MICROFICHAS

Fichero de programación en código máquina para ZX SPECTRUM.

Realización: Pedro Sudón Aguilar.
Diseño gráfico: Juan José Redondo.
Colaboran: Manuel Rozas y Santiago Revellado.

Este fichero consta de 208 fichas que se distribuyen de la siguiente forma:

Serie	Cantidad	Contenido
	1 (0)	Introducción.
G	34 (0 a 33)	Glosario.
T	16 (1 a 16)	Tablas de consulta.
I	68 (0 a 67)	Fichas de instrucciones.
M	52 (0 a 51)	Rutinas de la ROM.
R	37 (0 a 36)	Rutinas de utilidades.

Glosario (índice)

FE DE ERRATAS

Glosario

G-20: En la rutina de Cifrado de textos y programas, después de la instrucción DEC BC debe añadirse **INC HL**.

Instrucciones

I-0: Se ha omitido el código **m** que representa a cualquier registro r,(HL),(IX + d) e (IY + d).

Rutinas de la ROM

M-3: La lista de rutinas para introducir y extraer datos del stack del calculador está incompleta e incluye erróneamente SLICING. La lista completa aparece en la microficha M-44.

M-14: Tanto para PO-CHAR como para PR-ALL los datos de entrada y salida son:

Datos de entrada: B = 24-línea.
C = 33-columna.
HL = Direc. de esta posición.
A = Código del carácter.

Datos de salida : BC = Siguiente posición.
HL = Siguiente dirección.

M-17: La rutina CL-SCROLL tiene como dato de entrada: B = número de líneas.

M-20: La rutina KEY INPUT devuelve a la salida los siguientes flags:

Carry (C) = Código aceptable.
Zero (Z) ≠ No hay tecla pulsada.
NC y NZ = Código inaceptable (pulsación incorrecta).

Rutinas

R-0: El cargador hexadecimal no comprueba la última línea DATA, para que ello suceda deben cambiarse las siguientes líneas:

```
1030 LET Línea = 0 : LET Fin = 0
1100 IF n$(1) = " "THEN LET FIN = 1:GOTO 1150
1160 PRINT"LINEA ";Línea;" OK":IF NOT FIN
THEN GOTO 1050.
1165 PRINT "CARGA CORRECTA":STOP
```

Elimínense posteriormente las líneas 1220 y 2000.

Un ordenador es una estructura compleja capaz de realizar procesos en tiempos casi insignificantes, por medio de los cuales, a partir de unos datos conocidos, se obtienen las informaciones necesarias.

La *CPU (unidad central de proceso)* controla las operaciones, y la *memoria* proporciona el espacio para almacenar los datos, constituyendo en su conjunto lo que llamamos un ordenador.

Para que pueda funcionar un ordenador y sea útil, es preciso un soporte físico *(Hardware)* y un soporte lógico o *Software,* y para que las operaciones lleguen a realizarse, tienen que ser programados previamente mediante lenguajes familiares al usuario tales como *Basic, Ensamblador, Forth, Pascal, Logo, C, etc.*

Estructura Interna

La CPU (en nuestro caso el *Z80 A*) está compuesto para poder utilizar todas sus funciones, de *registros* (de propósito general y especiales) siendo los más significativos el puntero de pila o

Ordenador
CPU
Memoria
Lenguajes
Periféricos

Stack Pointer (SP), *contador de programa* o Program Counter (PC), *el registro de Flags* (F) y el *acumulador* (A).

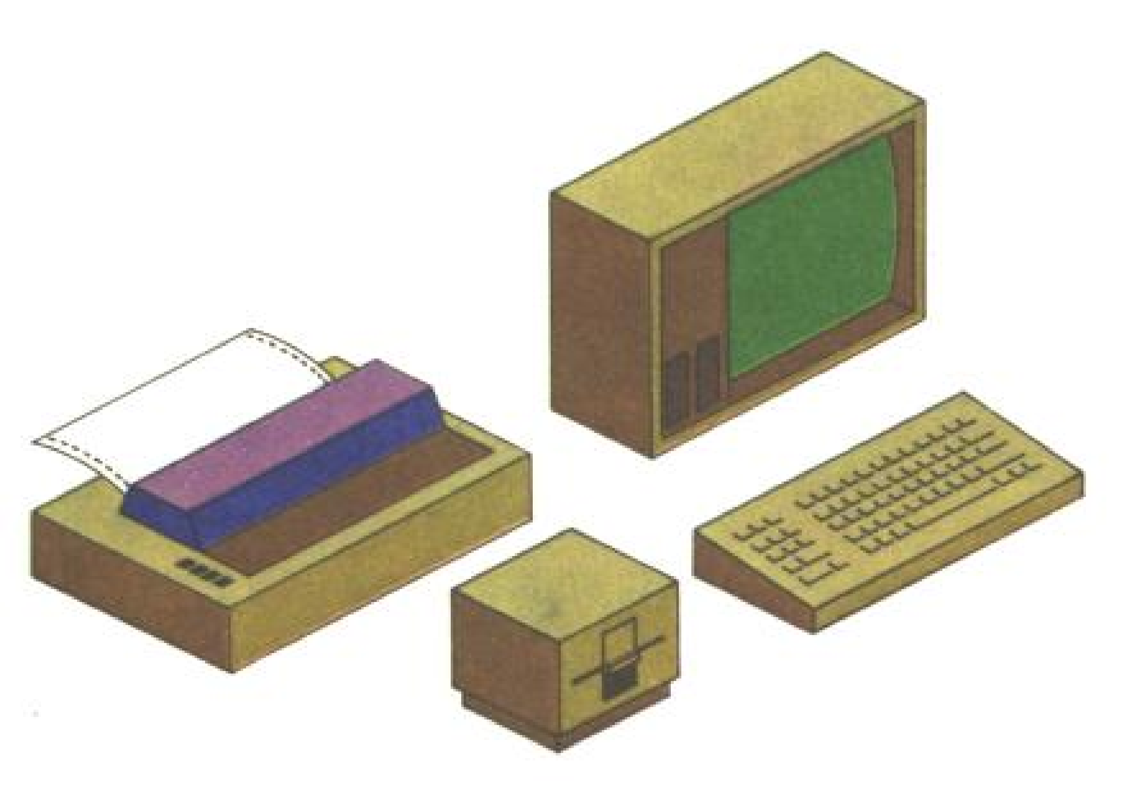
Lenguaje Ensamblador

Para comunicarnos con el computador lo hacemos mediante un *lenguaje* comprensible para el programador, pero la CPU no lo entiende, por lo tanto este lenguaje tiene que ser *traducido* dentro del mismo computador a *código máquina* para que sea comprendido.

Se pierde mucho tiempo en *interpretar* el Basic y lo ideal sería que nosotros aprendiésemos a hablarle en su propio lenguaje para ahorrarnos

tiempo; pero nosotros no podemos comunicarnos directamente con la CPU. Necesitamos un programa *ensamblador* para convertir las instrucciones que nosotros le indiquemos (en forma de *mnemónicos*) a lenguaje máquina.

Un programa ensamblador (a cuyo lenguaje de programación se le denomina también ensamblador y utiliza mnemónicos para crear código máquina), tiene la particularidad que puede facilitar la labor de programación con múltiples ayudas tales como *etiquetas, comentarios, pseudooperandos, etc.*



Interfaces/Periféricos

El ordenador se comunica con el usuario mediante los periféricos de entrada-salida *(input-output)* y de *almacenamiento*, que pueden tener a su vez su propio Hardware y su propio Software. Un ordenador se comunica con el periférico a través de un interface salvo algunos casos como son *cassette, TV y teclado*, que son los mínimos exigibles y no lo necesitan. Por lo tanto, lo que se conecta a los *ports* del computador es un *interface*, y a éste ya se le puede conectar el periférico.

Cada periférico tiene su interface *(Interface Centronics o RS232 para impresora, Interface 1 para Microdrive, interface para unidad de discos, joystick, lápiz óptico, vídeo, etc.).*

La Unidad Central de Procesos Z-80-A, creada por ZILOG en 1981 y fabricada actualmente por varias firmas con gran éxito comercial, es un circuito integrado de 40 patillas, y tiene como principales características:

- 158 microinstrucciones manteniendo compatibilidad con las 78 del anterior 8080A de Intel.
- Reloj rápido, a 4 MHzs.
- Juego amplio de registros internos (26 Bytes).
- Juego de instrucciones para el manejo de cadenas, bits, Bytes y palabras y para transferencia de bloques, con direccionamientos como el indexado y el relativo.
- 3 modos de interrupciones, según la compatibilidad necesaria con el Hardware de los periféricos.

Esta unidad en sí opera con 8 bits de datos, o sea, 1 Byte, que forma el llamado Bus de Datos, y en 16 bits para el Bus de Direcciones, pudiendo de esta manera direccionar 2 ↑ 16 (65536) posiciones de un Byte cada una (64 KBs.).

Descripción	Bus de direcciones
Características	Bus de control
Patillaje	Alimentación
Bus de datos	Reloj

PATILLAJE (Fig. 1).

Marcaremos las patillas del Bus de Datos con la letra D (Data-Bus), seguido de su orden de peso del 0 al 7, y las del Bus de Direcciones, con la letra A (Address-Bus), también con su peso del 0 al 15.

La dirección de la flecha indica:
Hacia fuera que es una patilla de salida.
Hacia dentro que es entrada.
Ambas direcciones que es bidireccional.

CLK > Clock o reloj de 4 MHzs.
+ 5 > 5 voltios de alimentación.
$\overline{\text{INT}}$ > Petición de interrupciones enmascarables (desautorizables).

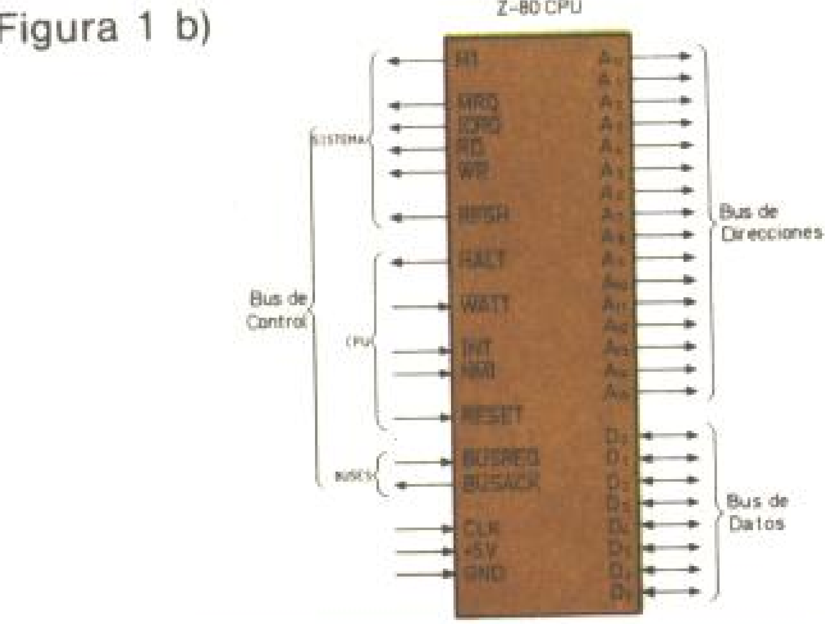
NMI	>	Petición de interrupción no enmascarable.
HALT	>	Indicación de parada de la CPU (espera de una interrupción para arrancar).
MREQ	>	Operación de direccionamiento a memoria.
IORQ	>	Idem/MREQ pero con periféricos (I/O).
RD	>	Bus de Datos en Entrada.
WR	>	Bus de Datos en Salida.
BUSAK	>	Disponible el acceso directo a memoria (DMA – Direct Access Memory).
WAIT	>	Espera de datos para transferencias lentas.
BUSRQ	>	Petición de DMA.
RESET	>	Puesta a 0 de la CPU.
M1	>	Primer ciclo de máquina
RFSH	>	Refresco de memorias dinámicas.
COMUN	>	Común de alimentación y señales. (0 Voltios)

(Figura 1 a)

Z-80 CPU

A_{11}	1	40	A_{10}
A_{12}	2	39	A_9
A_{13}	3	38	A_8
A_{14}	4	37	A_7
A_{15}	5	36	A_6
CLK	6	35	A_5
D_4	7	34	A_4
D_3	8	33	A_3
D_5	9	32	A_2
D_6	10	31	A_1
+5V	11	30	A_0
D_2	12	29	GND
D_7	13	28	RFSH
D_0	14	27	M1
D_1	15	26	RESET
INT	16	25	BUSREQ
NMI	17	24	WAIT
HALT	18	23	BUSACK
MREQ	19	22	WR
IORQ	20	21	RD

(Figura 1 b)



Z80A (Interior)

La Unidad Central de Proceso es el intelecto o cerebro, por así decirlo, del ordenador, que se encarga de realizar las operaciones aritméticas lógicas, de sincronización, de control y de la ejecución del programa, controlando todo el sistema.

Dentro de la CPU, al igual que en el resto del ordenador, los datos y señales de control se desplazan a través de los Buses, que son conjuntos de conductores eléctricos, a razón de un conductor por cada bit.

Tiene tres buses, uno interno para datos de 8 bits, otro para direcciones de 16 bits y otro de control de 13 bits, que sincroniza la CPU con el exterior.

La ALU (Aritmetic Logic Unit), o unidad logico-aritmética, se encarga de realizar las operaciones lógicas y aritméticas.

Los registros, que pueden almacenar un Byte, forman una pequeña memoria de uso interno de la CPU; son:

CPU
Bus de Datos
Bus de Direcciones
Bus de Control

La ALU
Registros
Funciones auxiliares

1. Registros de propósito general.

A, B, C, D, E, H y L; acumulador y registros de uso general (2 grupos).

IX e IY; registros dobles para direccionamiento indexado.

SP; registro doble que contiene la dirección actual de la pila de la CPU.

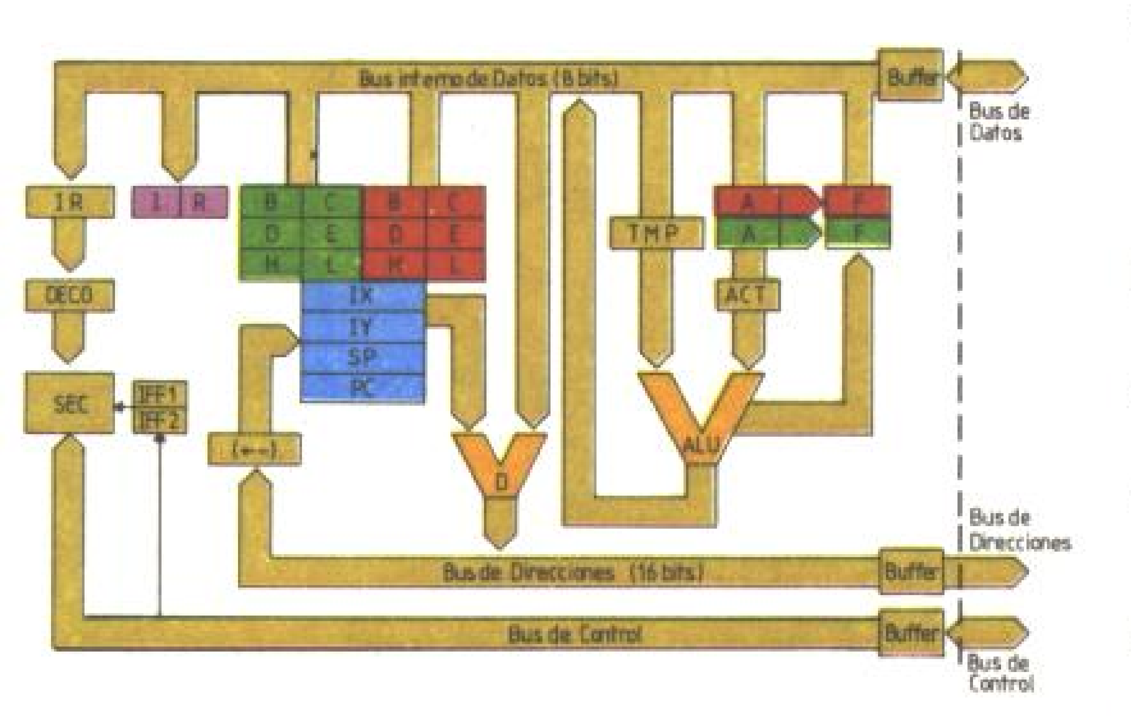
2. Registros indicadores de estado.

F; formado por los bits de condición (Flags o banderas), que son afectados por las operaciones; hay 2 registros F, uno por grupo de uso general.

I; registro que contiene el vector de interrupción en el modo IM 2.

R; registro contador para el refresco de memorias RAM dinámicas.

IFF1, IFF2; 2 bits indican petición de interrupción.



3. Registros de control de la CPU.

PC; registro doble que contiene la dirección de la instrucción que se está ejecutando.

IR; registro que contiene la instrucción que se está ejecutando.

TMP; registro temporal para operaciones.

ACT; acumulador temporal para operaciones.

Otros módulos, que realizan funciones auxiliares:

(+—); incrementador-decrementador de unidades.

D; operador de desplazamiento de direcciones.

DECO; decodificador de las instrucciones.

SEC; controlador de la secuencia de operaciones correspondientes a cada instrucción.

SALIDAS; para la adaptación de los Buses de la CPU con los Buses externos.

Un sistema de numeración es un convenio adoptado para expresar las cantidades mediante símbolos.

Estas cantidades se expresan en números que estarán formados por una cifra (o guarismo), o por una combinación de éstos, donde se tendrá en cuenta la posición que ocupan.

Se llama base al número de unidades de un orden que forman una unidad de orden superior (de peso mayor).

El peso es el valor representativo de cada posición dentro de un número, y se calcula elevando la base del sistema al ordinal de la posición menos 1: $p=b^{(n-1)}$.

Por lo tanto un número en cualquier sistema de numeración se puede expresar combinando las cifras que lo forman con los pesos correspondientes a cada posición.

- El sistema habitual de numeración es el decimal o en base 10, en que los números se forman a partir de 10 cifras diferentes.

Así, el número 249 está formado por las cifras 2, 4 y 9, y se podrá expresar como:

Sistema	Binario
Base	Hexadecimal
Peso	Notación
Decimal	Codificación

$$2 \cdot 10^2 + 4 \cdot 10^1 + 9 \cdot 10^0 =$$

$$2 \cdot 100 + 4 \cdot 10 + 9 \cdot 1 = 249$$

diremos que 1, 10 y 100 son los pesos correspondientes a la primera, segunda y tercera posición, 1 es el peso más baso o menos significativo, y 100 es el peso más alto o más significativo.

- El sistema de numeración que usan los ordenadores es el binario, debido a las limitaciones del propio hardware, que para garantizar una fiabilidad mínima sólo maneja bits, o números formados por 2 guarismos posibles, el 0 y el 1, siendo por lo tanto un sistema de numeración en base 2.

Siguiendo la misma lógica, el número binario 1001 equivale a:

$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$
$1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 9$

El sistema binario de los ordenadores no permite despreciar los ceros por la izquierda, aunque no tienen ningún valor, y existen convenios respecto del número de cifras o dígitos que pueden contener, habitualmente una potencia de 2 (4, 8, 16, 32).

- Puesto que el sistema binario utiliza bastantes dígitos, se suele emplear el sistema hexadecimal, o en base 16, por que cada cifra de éste representa 4 dígitos binarios.

Este sistema tiene 16 cifras posibles, que son del 0 al 9, y de la A a la F, lo que representa un rando del 0 al 15.

Por lo tanto, el número 7E en hexadecimal se puede expresar como:

$7 \cdot 16^1 + E \cdot 16^0 =$
$7 \cdot 16 + 14 \cdot 1 = 126$

— Se llama notación a la manera de escribir un número, y está generalmente aceptado que los números hexadecimales nunca empiezan por una letra (se añade un 0 al principio si es necesario), y se les añade una H al final, así como a los números binarios se les añade una B.

— Se llama codificación a la relación entre los números y su significado, formando una tabla de definiciones, que es la tabla de códigos.

Así, a cada instrucción de la CPU corresponde una serie de números, que se llama código de la operación, y a cada letra, en el código ASCII, le corresponde también un conjunto de números.

El microprocesador Z80 A tiene registros cuya característica es la de acceder a ellos para almacenamiento de **datos temporales** para poder realizar operaciones con ellos sin necesidad de utilizar memoria RAM externa. Existen dos juegos de registros de propósito general pudiéndose reservar un juego de ellos además del AF para el manejo de una rutina de acción inmediata.

1. El Acumulador
2. El par de registros HL
3. Los pares de registros BC y DE
4. Los registros indexados IX e IY
5. El puntero de pila o SP
6. Los registros especiales
   - Registro de banderas o Flags
   - Registro de interrupciones
   - Registro de refresco de memoria

1. El Acumulador:

Es el registro más utilizado ya que realiza y contiene el resultado de las operaciones lógicas y aritméticas con 8 bits. Las operaciones que pueden realizarse con el acumulador son: transferencias, suma, resta, AND y OR lógicos, XOR (or exclusivo lógico), comparaciones y complementación a 1 y a 2.

2. El par HL:

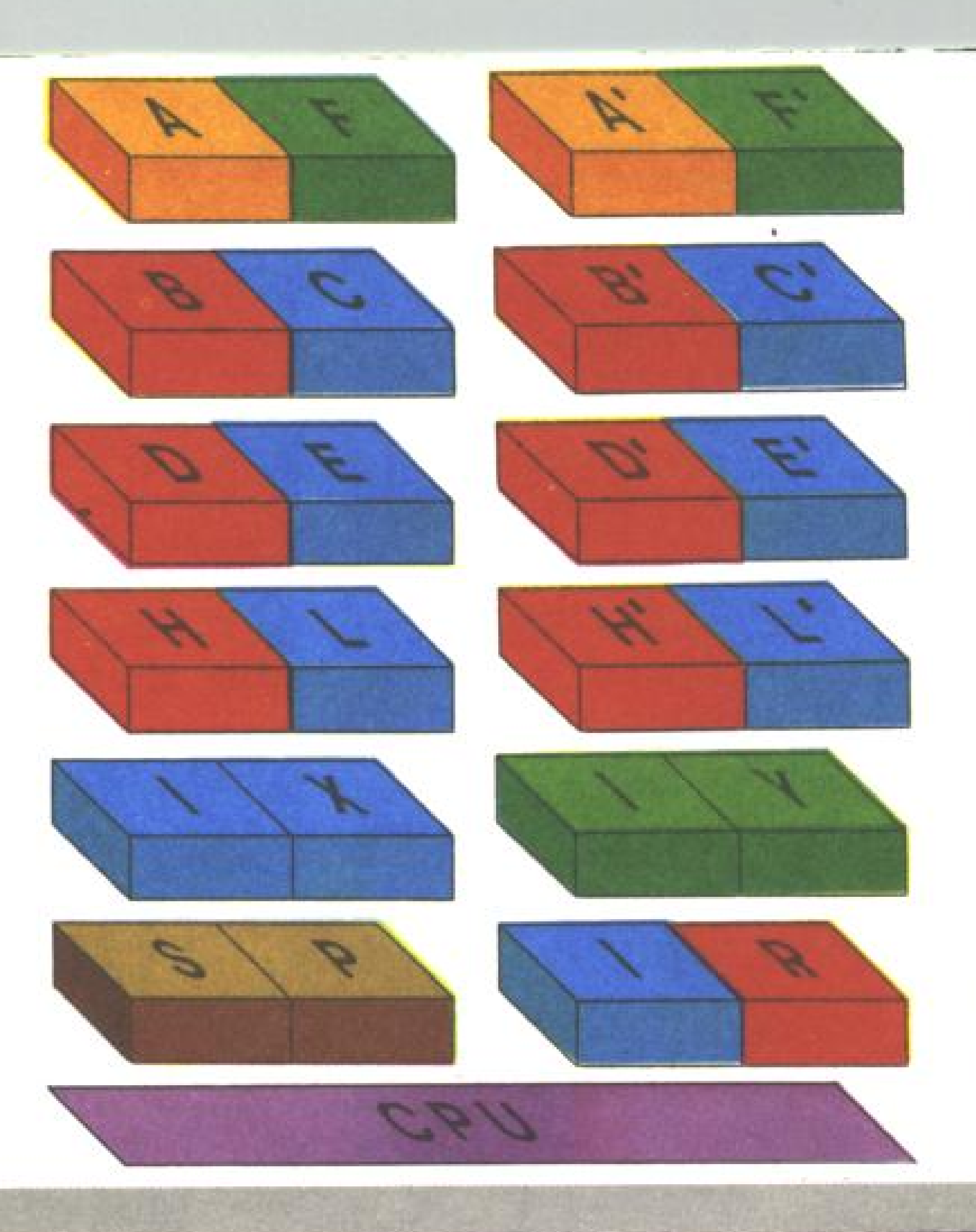
Es el par de registros más versátil de todos los que contiene el Z80 A, utilizado normalmente para contener las direcciones de memoria que se van utilizando durante el transcurso de una rutina, ya que algunas operaciones con los otros pares (BC y DE) no son ejecutables.

3. Los pares BC y DE:

Suelen utilizarse como pares auxiliares del HL en instrucciones que manipulan bloques tales como LDI, LDIR, etc.

4. Registros indexados IX e IY:

Los registros índice se utilizan como base para apuntar a una región de memoria de donde se va



a tomar o almacenar un dato. Se incluye un byte adicional para especificar un desplazamiento de esta base.

5. El puntero de pila SP:

La pila o stack está organizada de forma que el último dato que entra en la misma es el primero que sale. Esta organización permite el anidamiento ilimitado de rutinas.

6. Registros especiales:

— Registro de indicadores o Flags (F): indica las condiciones que se producen al realizar las operaciones en 8 y 16 bits.

— Registro de interrupciones I: Se utiliza para ejecutar cualquier subrutina como respuesta a una interrupción en modo IM2.

— Registro de refresco de memoria R: el dato del contador de refresco se coloca en la parte baja del bus de direcciones junto con una señal de control de refresco proporcionada por la CPU, mientras ésta busca y decodifica la instrucción.

La función USR

La función **USR** del Basic del ZX Spectrum es como el cordón umbilical que une el Basic en sí, con los programas escritos en código máquina.

Realiza además otra función, cuando el argumento es de tipo cadena, que nos da la dirección de comienzo de los caracteres **UDG** (Gráficos definibles por el usuario).

Con una expresión numérica, el BASIC hará una llamada a una subrutina en código máquina que comience en la dirección indicada por el valor de la expresión.

En la subrutina debemos preservar el par de registros IY, que es el puntero para las variables del sistema, y debe apuntar siempre a la variable **ERR-NR,** dirección 23610 (5C3AH).

Debemos también preservar el par de registros **HL',** que contiene información necesaria para el calculador del BASIC.

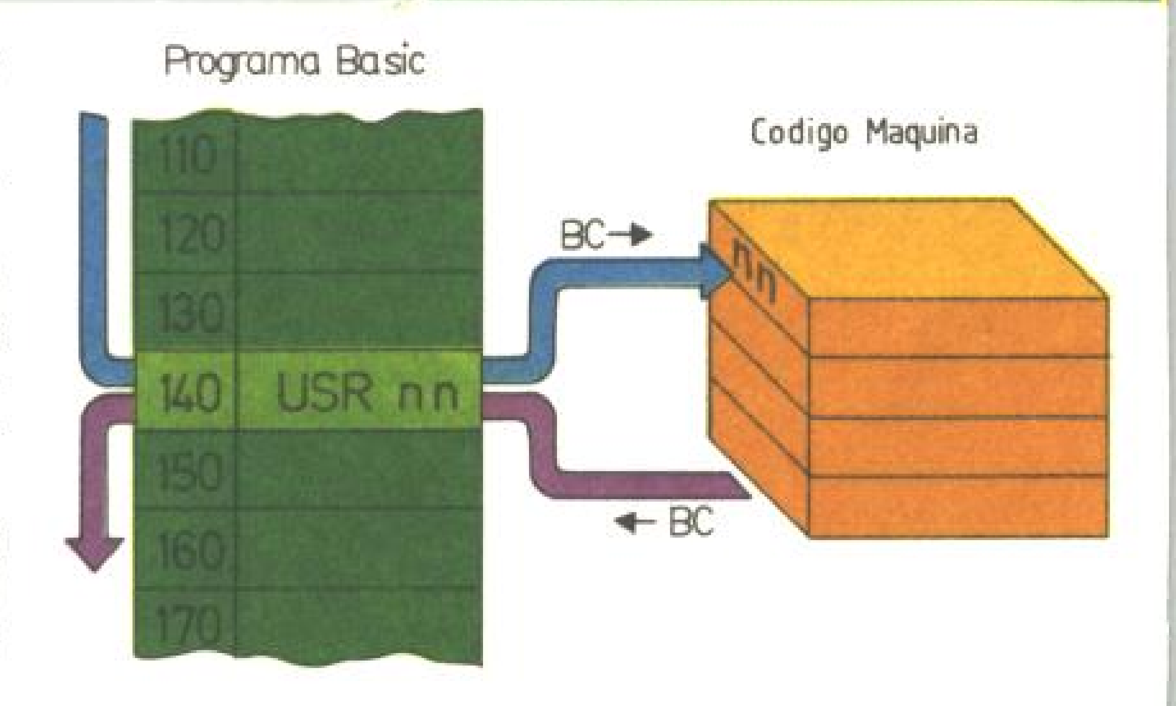
Podemos, además, conocer la dirección de comienzo de la subrutina, que está en el par de registros BC, dato necesario para reubicación y manejo de memoria.

Llamada a una subrutina en código máquina
Dirección de llamada
Parámetros numéricos con POKEs
Parámetros numéricos con REM
Parámetros numéricos en expresión
Valor de retorno

Por otra parte, la función USR devuelve el valor en decimal del par de registros **BC.,** muy útil para usar con variables númericas, por ejemplo, **LET num=USR nn,** donde se llama a una subrutina que comienza en la dirección nn, y al volver, la variable numérica «num» tiene el valor decimal del par **BC.**

Para pasar a su vez parámetros a la subrutina, podemos utilizar 3 sistemas:

- **POKE**ando los valores numéricos en las direcciones determinadas.
- Colocándolos en una instrucción **REM**, en la siguiente línea después de la función, que no provoca errores de sintaxis, cuya dirección de comienzo está en la variable del sistema **NEXTLIN**, dirección 23637 (5C55H).
- Usando USR en una expresión que conlleve el almacenamiento de los parámetros en el **Stack** del calculador BASIC, teniendo en cuenta la jerarquía de la expresión.
  Ej.: RANDOMIZE 1 + a * USR nn
  En este caso, el Basic chequea la expresión, y carga en el Stack los valores 1, y el de la variable «a», y antes de realizar las operaciones ejecuta la llamada al código máquina, puesto que por tener mayor prioridad, ha de realizar primero la multiplicación, en la que USR nn es el multiplicador, y el resultado final de la expresión se usará para el RANDOMIZE en sí, almacenándolo en la variable de sistema **SEED**, dirección 23670 (5C76H).



Cuando se trabaja con el **Interface 1**, sólo se puede utilizar con las instrucciones RANDOMIZE y LET, puesto que garantiza la correcta paginación de la ROM principal, contra otras instrucciones, especialmente IF USR nn, que pueden dejar el sistema completamente «colgado».

La mayoría de las instrucciones del Z80 operan sobre datos almacenados en los registros internos de la CPU, en la memoria externa o en los ports de entrada/salida.

La forma de generar la dirección de los datos para cada instrucción se denomina direccionamiento, pudiendo éste ser de los siguientes modos:

Directo

Cuando el código de operación incluye el operando al que se refiere la instrucción, es decir, operará directamente con el contenido de cualquier registro, o con cualquier operando numérico de 8 o 16 bits.

Indirecto

Cuando el operando en sí constituye una dirección de memoria, con cuyo contenido opera la instrucción.

En este modo el operando se escribe entre paréntesis y se lee «el contenido de».

Modos	El operando
Directo	Desplazamiento
Indirecto	

Indirecto Indexado

El byte siguiente al código de operación contiene un desplazamiento «d» implicito, que se suma a uno de los dos pares de índice, resultando la dirección de memoria donde se encuentra el operando.

Indirecto Relativo

El byte siguiente al código de operación especifica el desplazamiento «d» implicito, que ha de sumarse al contador de programa, ejecutando el salto correspondiente dentro del programa, de una manera semejante al modo indexado.

Según la naturaleza del operando puede ser:

Implicito

La instrucción indica, en su propio código de

	Implicito	Inmediato	Extendido	Pág. 0	bit
Directo	LD A,B	LD A,n	LD HL,nn	RST p	SET b,A
Indirecto	LD A, (HL)	LD (HL),n	LD (HL),nn	—	SET b,(HL)
Indexado	LD A, (IX+d)	LD (IX+d),n	—	—	SET b,(IX+d)
Relativo	JR d	—	—	—	—

operación, el operando que maneja, habitualmente registros o indicadores de condición.

Inmediato

El byte siguiente al código de operación de la instrucción es el operando (de 8 bits).

Inmediato Extendido

El operando (de 16 bits) son los dos bytes siguientes al de código de operación, el primero es el byte bajo (Low) o menos significativo, y el segundo, el byte alto (High) o byte más significativo.

Modificado a página 0

El código de operación de la instruccion determina cualquiera de las 8 posibles direcciones de llamada en la instrucción RST, situadas en la página 0.

La página 0 es la primera porción de 256 bytes de la memoria.

De bit

El código de operación de la instrucción especifica cualquiera de los 8 bits de un byte.

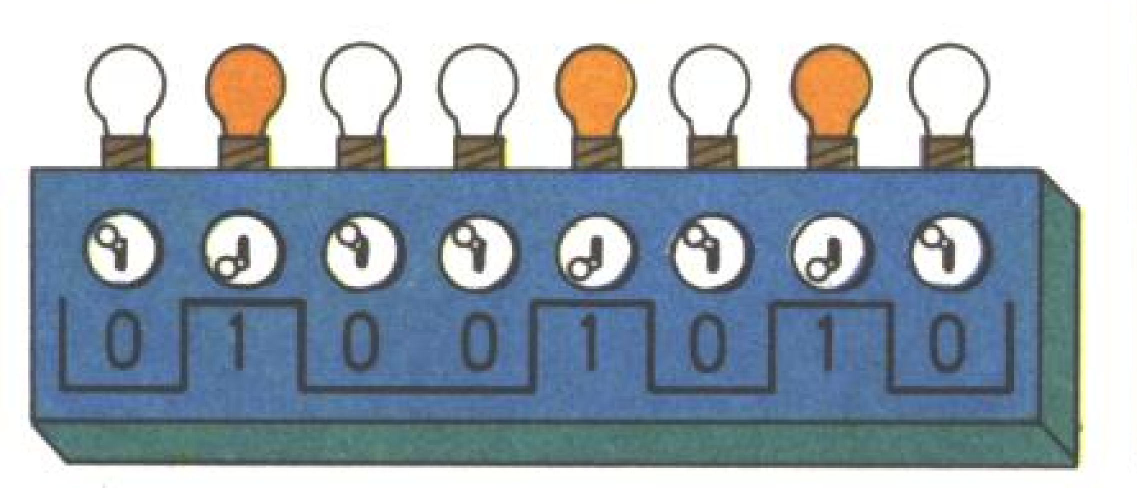
• En los modos relativo e indexado, el desplazamiento «d» lo constituye un byte que se interpreta como complemento a 2, que cambia el rango ordinario de 0 a 255 por el rango con signo, que comprende de 0 a +127 y de 0 a −128.

El ordenador utiliza el sistema en Base 2 para su funcionamiento:

Bit:

La palabra bit, abreviatura de binary digit, dígito binario, es como una bombilla mandada por un interruptor, que, o está encendida, o está apagada.

El origen de esta palabra está en cómo funciona un ordenador por dentro; cada conducto eléctrico, independientemente, puede tener tensión o no, lo que en términos de lógica algebraica se llama verdadero o falso, en hardware alto y bajo, y en informática 1 ó 0.



Bit (binary digit)

0	1
bajo (low)	alto (high)
falso (false)	verdadero (true)

Palabra (word) (conjunto de bits)

1
4
8 (Byte, Octeto)
16 (Palabra de la Z80)
20
32

Record (conjunto de Bytes dividido en campos)

128
256
512
1024

Palabra:

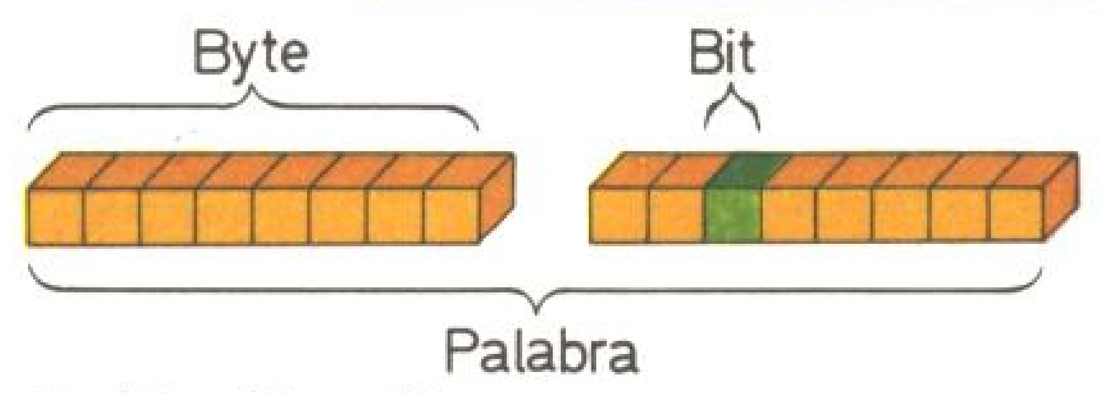
Se llama palabra (word), al conjunto de bits que unitariamente tienen un significado concreto para el ordenador, y que a su vez pueden ser manejados en conjunto.

El tamaño viene determinado inicialmente por el propio hardware del ordenador, y normalmente es un número potencia de 2, o al menos un número par (las palabras más usuales son de 1, 4, 8, 16, 20, o 32 bits).

Byte:

De etimología inglesa, al igual que octeto, de origen francés, significa una palabra de 8 bits, que es la más utilizada actualmente en informática.

En el caso del ZX Spectrum, donde la palabra de Datos es de 8 bits, y la palabra de Direcciones es de 16 bits, los usos prácticos aconsejan llamar Byte al Dato, y Palabra a la Dirección, términos aceptados por la gran mayoría de especialistas en código máquina del Z80.



Registro (Record):

Unidad lógica de información, es un bloque completo de información que se maneja todo a la vez (no confundir con los registros de la CPU).

Suele estar asignado a un Buffer, que es donde se aloja provisionalmente, para transacciones con los periféricos.

Los tamaños habituales para un registro son 128, 256, 512 o 1024 Bytes, que puede resultar grande, pero se puede seccionar en campos, siendo una pieza fundamental en el tratamiento de la información.

Así, por ejemplo, el registro de los ZX Microdrives es de 512 Bytes, y el registro de los discos flexibles (Floppy disk) es de 256 Bytes, habitualmente.

Un ensamblador es una herramienta de software (un programa), diseñado para simplificar las tareas que conlleva escribir los programas en código máquina, bien en binario o en hexadecimal.

El lenguaje ensamblador es una serie de nombres simbólicos (mnemónicos) de operación, fácilmente comprensibles, que se corresponden con las microinstrucciones de la CPU (Unidad Central de Proceso), lo cual obliga al programador de lenguaje ensamblador a conocer detalladamente cada una de las operaciones que ésta realiza.

Para usar el lenguaje ensamblador necesitamos un fichero de código fuente, que es una lista de líneas de texto, que deben cumplir las siguientes exigencias:

1. Número de línea, por cuyo orden son colocadas y ensambladas, a semejanza del Basic.

2. Campo de etiqueta, referencia necesaria para que el ensamblador desarrolle el flujo de programa deseado, en saltos u otras instrucciones que manejen direcciones.

Código fuente
Código objeto
Código máquina
Líneas de ensamblador
Campos
Ensamblaje en 2 pasos

3. Campo de código de operación (mnemónico), es opcional, y puede contener en lugar del código una directiva de ensamblador (pseudoinstrucción).

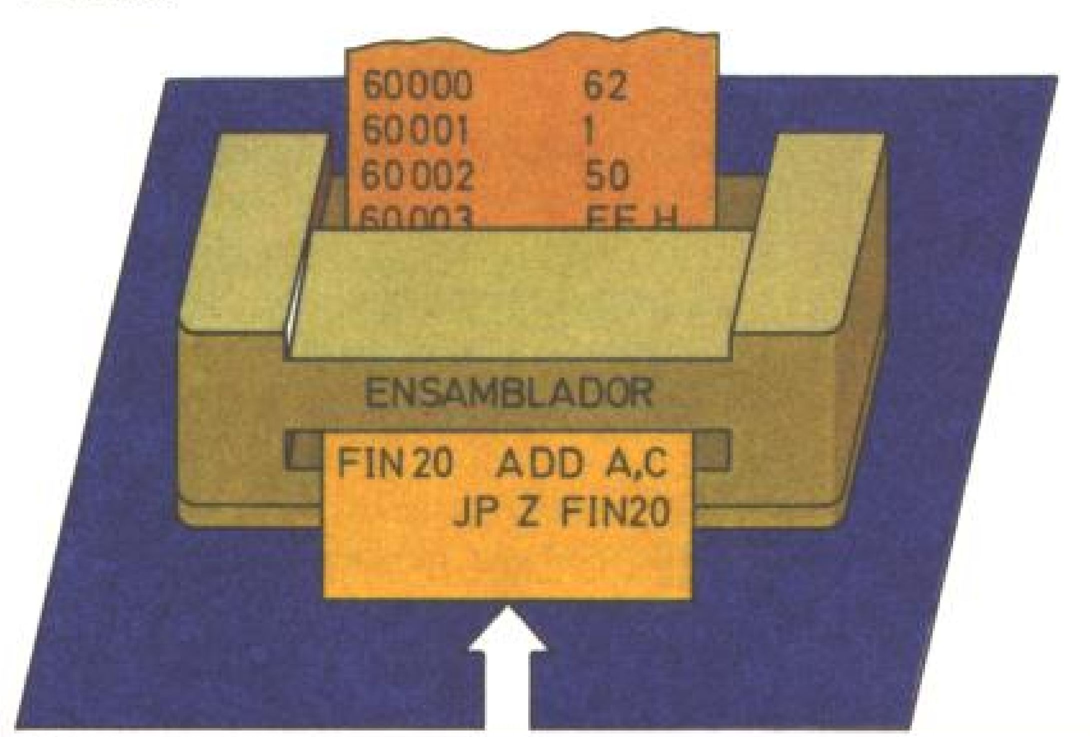
4. Campo de operando, también opcional, respetando la estructura del código mnemónico, puede tener ningún, uno o dos operandos, en este último caso deben ir separados por coma, y siempre que sean numéricos, pueden ser sustituidos por una expresión simbólica (con etiquetas).

5. Campo de comentario, opcional, de ayuda para entender mejor los programas, debe ir precedido de un punto y coma.

Todos los campos de una línea deben estar separados al menos por un espacio, siendo acon-

sejable el empleo de tabulaciones, para que queden alineados por columnas, que contribuye al mejor entendimiento del programa.

- Una expresión numérica en lenguaje ensamblador es una combinación de números, símbolos y operadores, respetando las reglas algebraicas, donde cada elemento de la expresión es un término, y el resultado debe estar acorde con el operando a que sustituye, en su rango, de 8 a 16 bits.



Normalmente una expresión numérica debe poder admitir números en cualesquiera de las bases corrientemente utilizadas en lenguaje ensamblador, o sea, binario, octal, decimal o hexadecimal.

Una vez tenemos el código fuente, podemos ensamblarlo, en dos pasos, para producir el código objeto.

- En ensambladores más potentes, normalmente con ordenadores de mayor tamaño, el fichero de código objeto se combina con otros ficheros para generar el código máquina, y en ensambladores más sencillos, este constituye directamente el propio código máquina, que es el ejecutable por la CPU.

En el primer paso se comprueban errores de sintaxis, errores de organización de memoria, y se calculan el espacio necesario y los desplazamientos de las direcciones relativas.

En el segundo paso, si no ha habido errores, se cumplimenta el código objeto, chequeando que los valores de los operandos estén en su rango, y las etiquetas estén en su lugar correcto (no haya etiquetas repetidas o inexistentes).

Reubicar

Una rutina es **reubicable** cuando se puede situar en cualquier dirección de la RAM disponible, sin que la misma deje de ser apta para la utilización; en otras palabras, es reubicable si, sea cual sea la dirección donde se sitúe, funciona sin dar ningún tipo de **error;** en caso contrario se considerará que no es reubicable.

Para saber si una rutina es reubicable hay que saber si tiene alguna instrucción CALL (llamada a subrutina), JP (salto) u otra cualquiera que se refiera de modo absoluto a una dirección que pertenezca a la rutina, en cuyo caso no es reubicable mientras no se le añada alguno de los sistemas de reubicación.

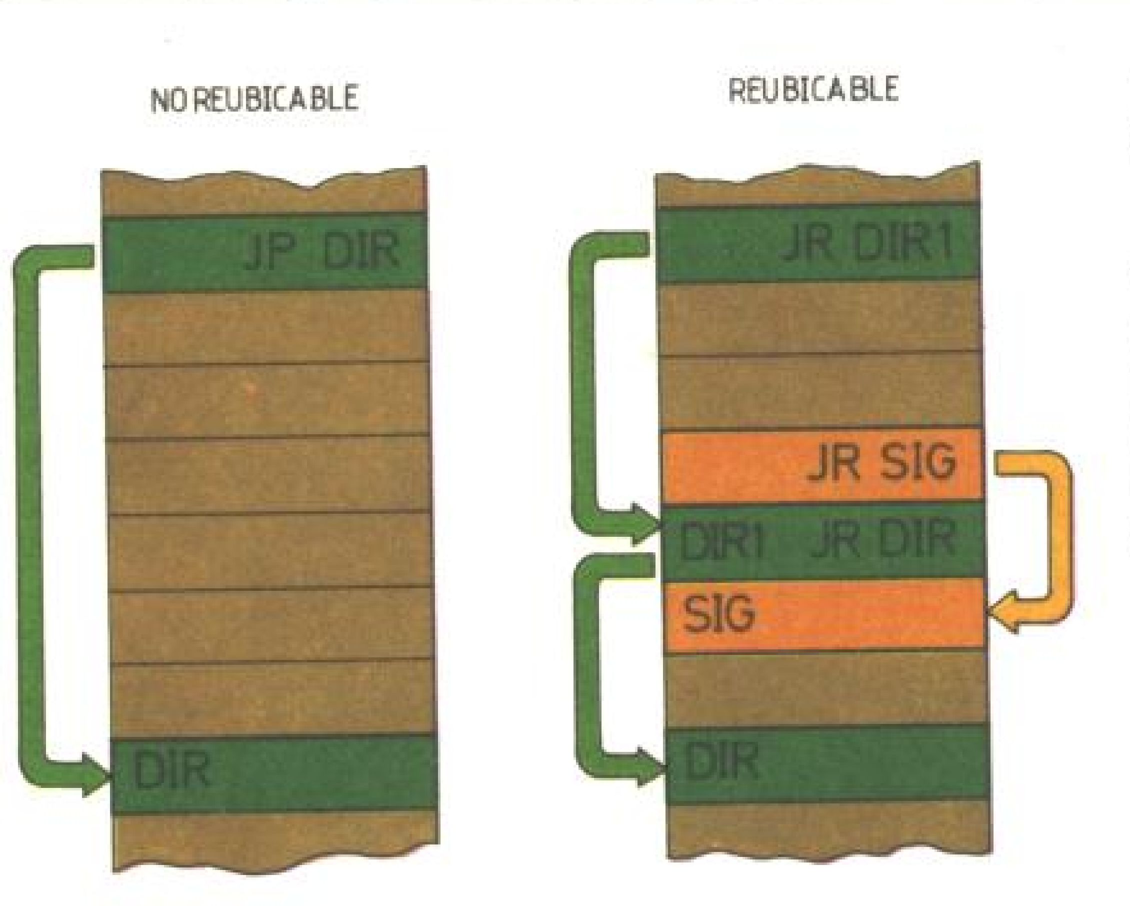
Así, cualquier relación con las direcciones de la ROM, de los ficheros de pantalla o de las variables del sistema no afectará de ninguna manera para que la rutina funcione correctamente, en cualquier posición de memoria.

Formas de hacer reubicable una rutina:

Un **JP (Salto absoluto)** que anule la posibilidad de reubicación de una rutina podrá ser sustituido por un **JR (salto relativo)** siempre que el salto en sí sea de 127 posiciones hacia adelante o 128 hacia atrás (como máximo).

Se puede sustituir un JP (Salto absoluto) de más de 128 posiciones por varios JRs (Saltos relativos) encadenados, que realicen la misma función, aunque provocan un retardo del tiempo de ejecución y ocupan mayor espacio de memoria. (Ver figura.)

- Concepto de reubicación (relocation)
- Características de las rutinas reubicables.
- Formas de hacer reubicable una rutina:
  - JR
  - Repetición de las subrutinas
  - Subrutina para sustituir CALL



● Se puede evitar un CALL (llamada a dirección absoluta), escribiendo la subrutina en lugar de los CALLs (llamadas) que la usen; de esta manera disminuirá ligeramente el tiempo de ejecución, pero ocupará más memoria.

El mejor método es ejecutar un trozo inicial de la rutina, cuya misión sea calcular las nuevas direcciones no relativas de la propia rutina.

También un **CALL (llamada dirección absoluta)** se puede sustituir por un JR (salto relativo), con los límites de direccionamiento señalados, si previamente las últimas instrucciones ejecutadas han actuado sobre la pila a través del par de registros **SP (Stack Pointer),** para apilar la dirección de retorno; así:

```
CALL    28
DEC     SP
DEC     SP          Equivale a:
POP     DE
LD      HL,10       CALL SUBRT
ADD     HL,DE
PUSH    HL
JR      SUBRT
```

Las etiquetas son nombres simbólicos, que pueden estar compuestos por letras, o por letras y números, pero siempre comenzando por una letra, a los que se les asigna un valor numérico, normalmente una dirección de memoria.

Son equivalentes a las variables numéricas del BASIC, por poner un ejemplo, primero hay que darles un valor, crearlas, y luego las usamos en representación de ese valor que así, es variable.

Por otro lado son parecidas a los números de línea del Basic, y sirven para calcular las direcciones de los saltos en código máquina.

Las etiquetas son siempre opcionales, siendo necesario respetar su lugar al comienzo de la línea de esamblador, seguida del separador, normalmente un espacio, antes de escribir el llamado símbolo mnemónico.

Hay dos formas de crearlas (declararlas):

1. De modo absoluto mediante EQU.
2. De modo relativo, tomando el valor del puntero de dirección.

Los nombres simbólicos como variables
Modo absoluto con EQU para expresiones numéricas
Modo relativo para direcciones del programa

El primer paso que realiza un ensamblador es producir un código máquina provisional, donde los valores numéricos que no están declarados absolutamente, sino que tienen una etiqueta, son considerados 0, y por otro lado, se asignan los valores correspondientes a las etiquetas, creando una tabla de correspondencia entre éstas y los valores calculados, que se llama tabla de símbolos.

En un segundo paso se asignan los valores de la **tabla de símbolos** al código máquina, reemplazando los 0 provisionales.

Ejemplo:

```
        ENSAMBLADOR                          C M

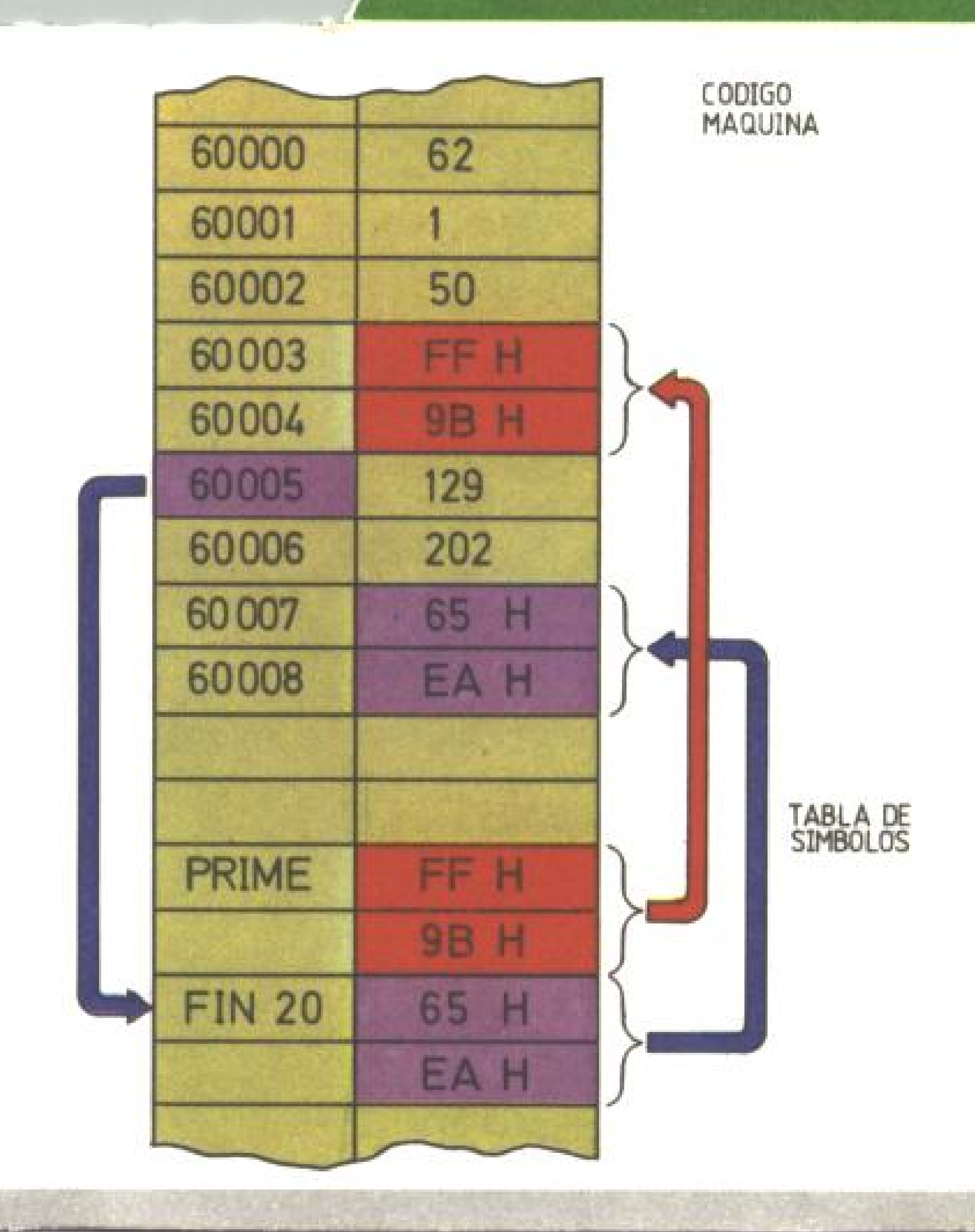
10          ORG   60000
20 PRIME    EQU   9BFFH
30          LD    A,1          60000   62,1
40          LD    (PRIME),A    60002   50,FFH,9BH
50 FIN20    ADD   A,C          60005   129
60          JP    Z,FIN20      60006   202,65H,EAH
```

En la línea 20, la etiqueta PRIME toma el valor 9BFFH (ejemplo de modo absoluto).

En la línea 50, la etiqueta FIN20 toma el valor de la dirección ADD, que sabemos que es 60005 (ejemplo de modo relativo).

Así LD (PRIME),A equivale a decir LD (9BFFH),A y de la misma manera JR Z,FIN20, es lo mismo que JR Z,60005.

EL utilizar FIN20 en lugar de 60005, tiene la ventaja de que si insertamos más instrucciones entre las líneas 50 y 60, la etiqueta FIN20 volverá a ser calculada por el ensamblador, por esto se llama modo relativo.



El registro f (flags) contiene los bits de prueba de condición, que son directamente consultados en las operaciones condicionales, no puede ser manipulado como un registro de propósito general, excepto a través de la secuencia PUSH AF y POP dd, que hace que el contenido de este registro se transfiera a la parte baja del par dd.

- Flags de uso general:
  Acarreo
  Paridad/Desbordamiento
  Cero
  Signo
- Flags de uso interno:
  Sustracción
  Medio acarreo

Bits que contiene:

0-C (acarreo)

El bit de acarreo del acumulador puede considerarse el noveno bit del mismo; se ve afectado por la ejecución de operaciones lógicas o aritméticas, u otras que lo usen explícitamente.

2-P/V (paridad/desbordamiento)

Puesto a 1 indica que el resultado de una operación lógica tiene paridad impar, o que el resultado de una operación aritmética en complemento a 2 ha producido desbordamiento.

6-Z (cero)

Puesto a 1 en instrucciones tales como comparaciones, rotaciones e instrucciones BIT, IN y OUT indica que el acumulador contiene cero.

7-S (signo)

Puesto a 1 indica que el resultado de una operación aritmética es negativa (es copia este bit del bit 7 del acumulador).

• Hay otros dos bits situados en el registro F no utilizables en saltos condicionales pero que sí se utilizan en aritmética BCD:

1-N (sustracción)

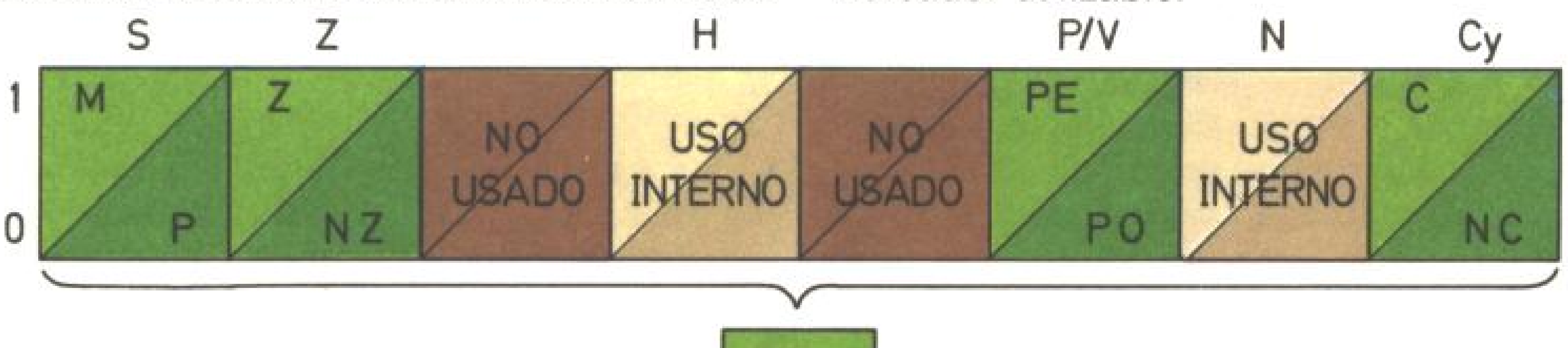
Puesto que el algoritmo para corregir operaciones BCD es diferente para sumas que para restas, este indicador indica a la CPU qué tipo de instrucción se ejecutó previamente de forma que la operación DAA efectuara la corrección adecuada en el resultado tanto de la adición como de la substracción.

4-H (Medio acarreo)

Es el acarreo de BCD generado a partir de los cuatro bits menos significativos, para indicar que han rebasado el valor 9.

Cuando se utiliza la instrucción de ajuste decimal (DAA) este indicador se utiliza para corregir el resultado binario a BCD.

• Los bits **3** y **5** no representan ningún tipo de indicador utilizable.





Para la confección de un programa lo primero que se debe hacer es la representación gráfica de la estructura lógica y operacional de los procesos del ordenador, y puede ser:

Funcional:

Muestra las grandes etapas de transformación que sufre la información sin referirse a ningún elemento del ordenador.

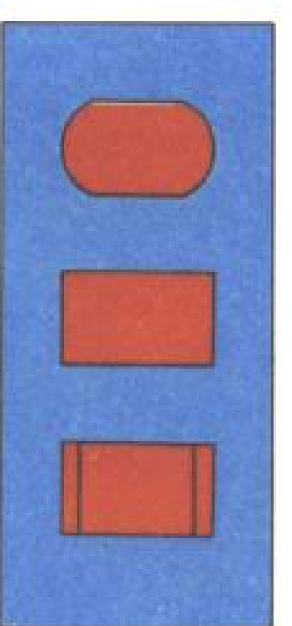
De procesos:

Se diferencia del anterior en que tiene en cuenta los elementos que constituyen el ordenador.

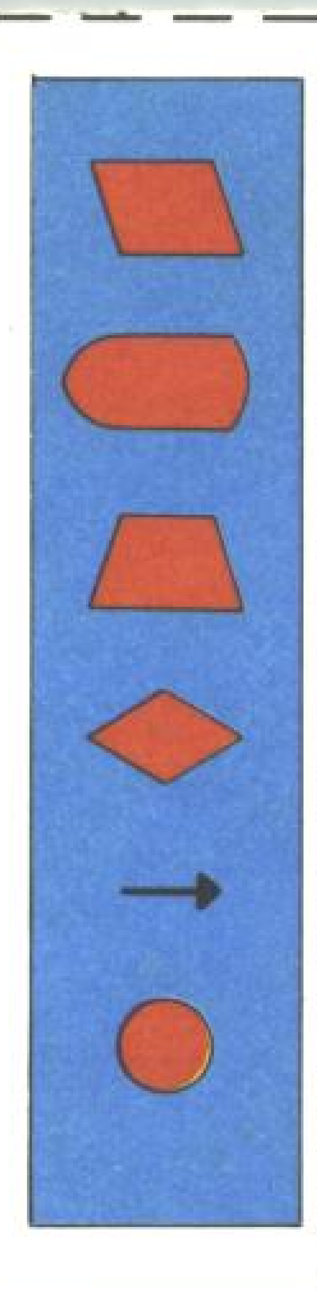
Ordinograma:

Recoge, gráficamente, todas las órdenes que en secuencia debe dar el hombre al ordenador para la solución del problema.

Simbología:



— **Terminal**
Principio, fin o cualquier tipo de salida del programa.
— **Proceso** (rectángulo)
Cualquier modo de operación que puede asignar cambio de valor, formato o posición de la información en la memoria.
— **Subrutina** (rectángulo barrado)
Llamada a una subrutina cuyo nombre se situará dentro del rectángulo.



Entrada/salida (romboide)
Transferencia de datos entre el sistema y los elementos periféricos; si es desde el sistema será salida y si es hacia el sistema será entrada.

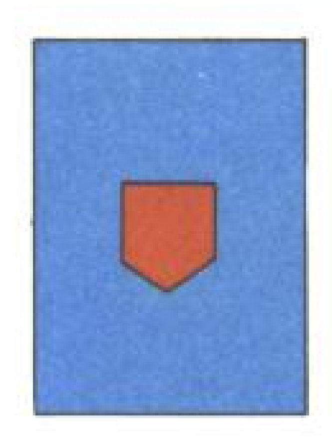
–Salida por pantalla
Transferencia del sistema a un monitor de vídeo.

– **Entrada Manual** (trapecio)
Entrada desde el teclado.

– **Decisión** (rombo)
Establece la comparación entre dos datos y en función del resultado determina cuál de los distintos caminos del programa debe seguir.

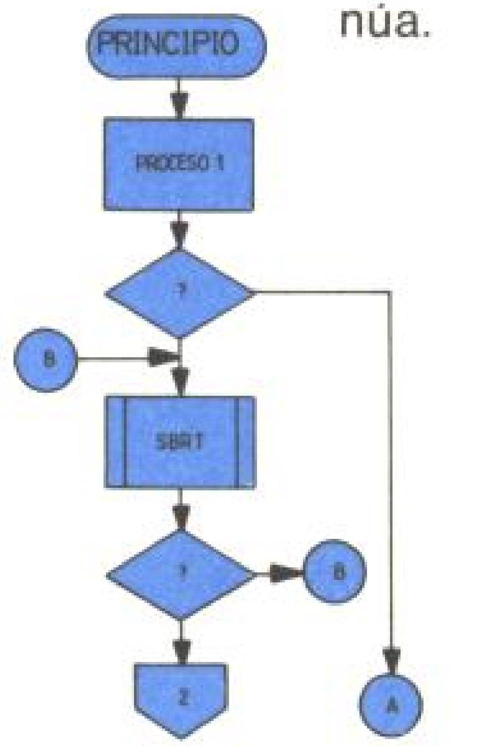
– **Línea de flujo** (flecha).
Indica la dirección de encadenamiento de los distintos símbolos.

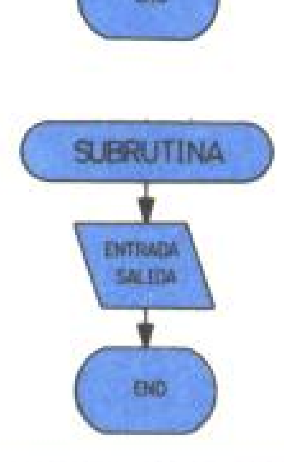
– **Conector** (círculo)
Enlaza dos partes del ordinograma, a través de un conector en el origen y un conector en el destino.



Ambos círculos deben contener una referencia o nombre de conexión.

– **Conector de página** (pentágono)
Conecta todas las páginas que sean necesarias para representar un ordinograma. Debe contener el número de página en que continúa.





Definición	Rango
Partes	Anidación

Un bucle es un bloque de instrucciones que tienen la particularidad de que controlan un mismo proceso repetidas veces.

Esto supone una gran simplificación del proceso durante la ejecución de un programa permitiendo que éste sea cíclico y esté perfectamente estructurado.

Además se acortan, el tiempo de ejecución, y el espacio que ocupa el programa.

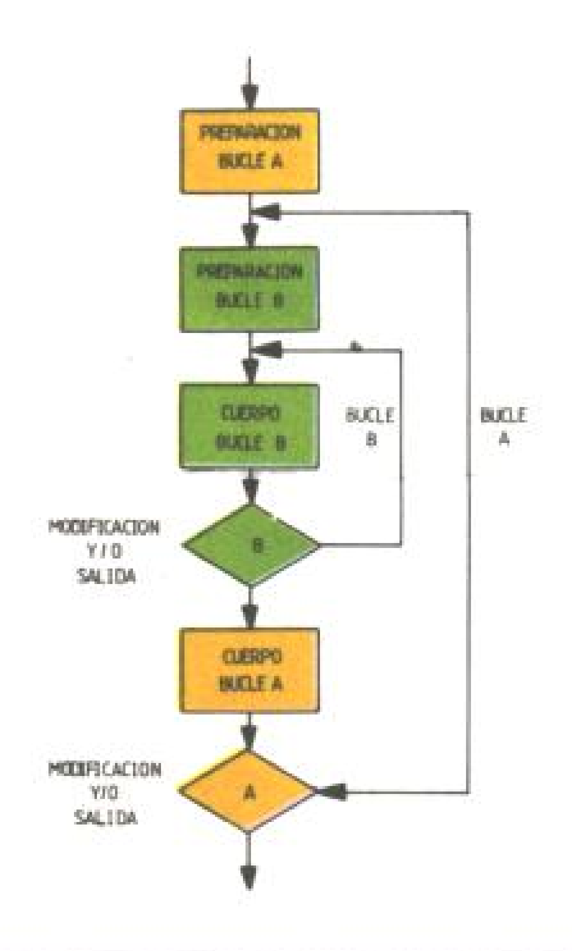
- Las operaciones en bucle constan de cuatro partes esenciales:

  1. Una o más instrucciones que sirven de **preparación** o arranque del bucle.
  2. Un grupo de instrucciones que constituyen el **cuerpo** del bucle y que se ejecutan repetidas veces.
  3. Un grupo de instrucciones que modifican el bucle haciéndole **progresar.**
  4. Una instrucción de **comprobación de salida** del bucle que sirve para investigar si se ha producido la condición que determina la salida del bucle. Si ésta no se produce, entonces continúa el bucle.

- La terminación del bucle puede realizarse de distintas maneras:
  - Cuando el índice alcanza el valor final.
  - Por cumplir una condición que modifica el proceso, saltando a un punto exterior al bucle.

- Puede convenir que la última sentencia de un bucle sea común a varios bucles diferentes, o bien que se realice un salto al interior de un bucle desde fuera de su rango. Debe tenerse cuidado en el diseño de este tipo de estructuras ya que debido a su complejidad existe el riesgo de producir errores.

- Se llama **anidación** de bucles cuando un bucle contiene dentro de su rango sentencias que forman otro bucle, el cual será considerado de menor rango, por ser interior:




```
        LD      C,na
BUCA    LD      B,nb
BUCB    proceso b
        DJNZ    BUCB
        proceso a
        DEC     C
        JP      NZ, BUCA
        continúa
```

El proceso «b», dentro del bucle BUCB, está anidado en el bucle BUCA, el cual además incluye el proceso «a».
Si estos procesos no afectan el desarrollo de los respectivos bucles, el proceso «b» se repetirá «nb» veces, cada vez que se ejecute el bucle BUCA, («na» veces).
También el proceso «a» se repetirá «na» veces, puesto que está incluido en el bucle BUCA.

Subrutinas

Programa principal	RET
Subrutina	Anidación
CALL	Encadenamiento

Dentro de un programa que efectúa un proceso definido, suele haber operaciones específicas que deben realizarse repetidas veces, y en cualquier punto de dicho proceso.

Entonces diferenciaremos dentro del programa el bloque principal, llamado **programa principal,** dentro del cual, y en cualquier punto de éste, podrán escribirse instrucciones de llamada (CALL o GOSUB) a otras partes del programa.

En los bloques de instrucciones que pueden ser llamados, denominados subprogramas o subrutinas, se incluirán las correspondientes instrucciones de retorno (RETURN o RET) al punto donde se produjo la llamada.

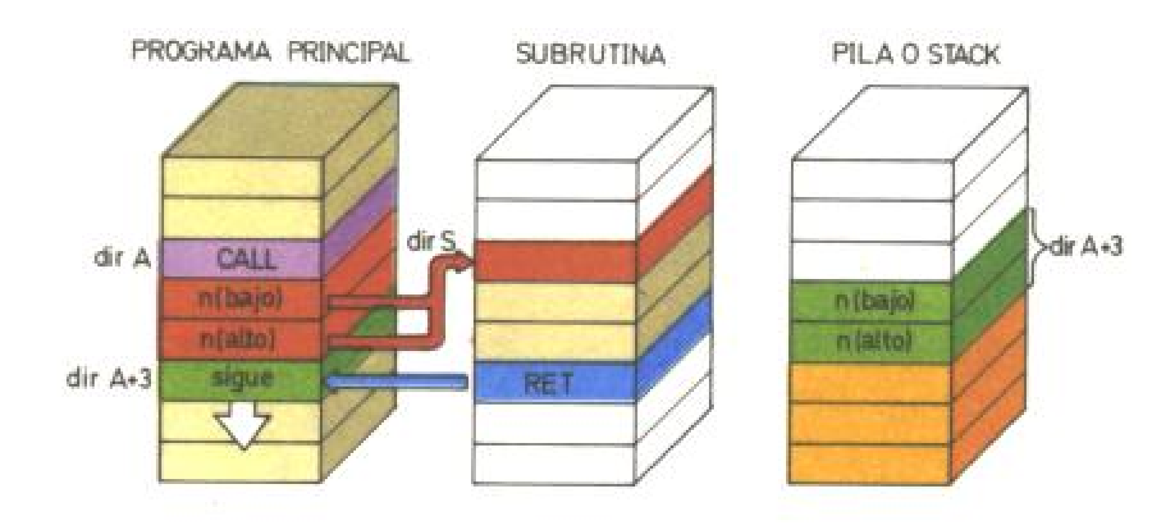
La CPU dispone de dos instrucciones específicas para el tratamiento de las subrutinas:

- **CALL nn**
  Equivale a decir salta a la subrutina que está en la dirección nn, guardando la dirección donde continúa el proceso en la pila de máquina, para que una vez terminada su ejecución pueda volver a este punto (Sería como PUSH PC + JP nn).
- **RET**
  Equivale a decir: Toma la dirección de retorno de la pila de máquina, y salta a ella, para continuar el proceso principal (Sería como POP PC).

Mediante este sistema, basta con tener una reserva suficiente de espacio para la pila de máquina, para usar todos los niveles que se deseen de subrutina.

Este es el concepto de **anidación,** esto es, el programa principal puede llamar a una subrutina en cualquier punto de éste, la cual puede llamar a su vez a otra subrutina, etc.

Por lo tanto, la pila de máquina debe ser cuidadosamente utilizada para no alterar las direcciones de retorno con los posibles datos temporales que use la subrutina.



- Se puede utilizar el siguiente método para **encadenar** subrutinas:
  - La subrutina sbrtA debe realizar el proceso A.
  - La subrutina sbrtB debe realizar los procesos B y A por este orden.

Entonces podremos escribir:

```
sbrtB    Proceso B
         JP sbrtA
sbrtA    Proceso A
         RET
```

- Si llamamos a la subrutina sbrtA, se efectúa el proceso A, y a continuación se efectúa el retorno (RET) al programa principal.
- Si llamamos a la subrutina B, se efectuará el proceso B, y mediante el salto JP se efectuará también el proceso A, que termina en el retorno (RET) al programa principal.
- Si la subrutina A está a continuación de la subrutina B, no es necesario el salto JP, ya que el flujo continuará en ésta directamente.

La memoria es el almacén de los datos en un ordenador, constituyendo un espacio físico y limitado, con una serie de características, normalmente conocidas, por las cuales se pueden dividir en tipos.

Las características principales de una memoria son:

- **Tamaño**
  La capacidad en bytes (Kilobytes o Megabytes).
- **Tecnología**
  Puede ser digital, magnética u óptica.
- **Método de acceso**
  Aleatorio por dirección de memoria (Byte a Byte), secuencial por bloque (acceso al siguiente bloque), o aleatorio a bloque (acceso al bloque deseado).
- **Velocidad de acceso**
  El tiempo que tarda en accederse a una posición.

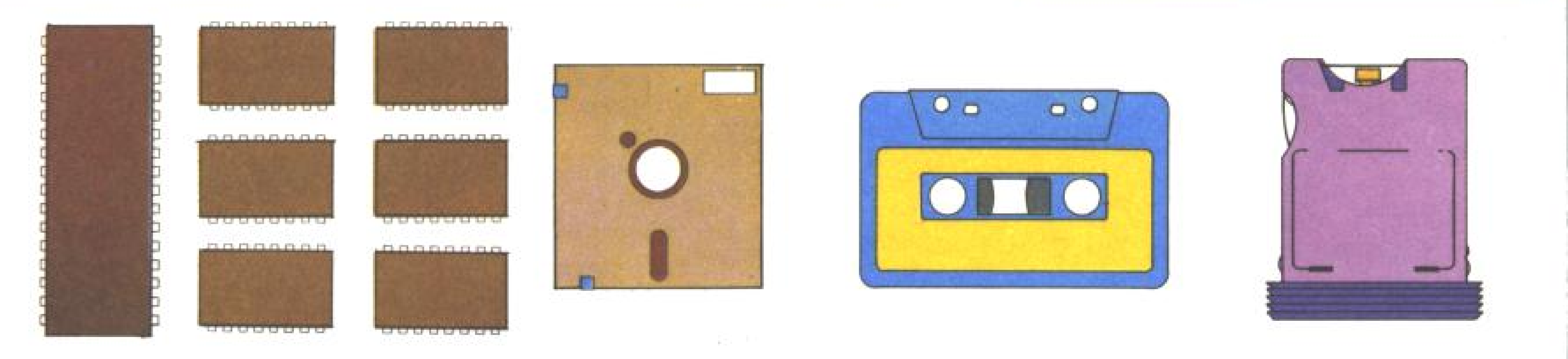
Características

Memoria Central	**Memoria de Línea**
RAM	Cassette
ROM	Microdrive
	Discos

- **Velocidad de transferencia**
  El tiempo que tarda en entrar o salir un dato.

Según esto, habrá 2 tipos genéricos de memoria:

- **Memoria Central**
  La usada por el procesador propiamente dicho, debe ser de acceso aleatorio, y de alta velocidad, con lo que suelen ser de pequeño tamaño:
  - **RAM** (Random Access Memory), memoria de acceso aleatorio, digital, velocidad rápida, tamaño pequeño (1 a 16



Kbytes), es temporal, ya que al quitarle la alimentación se borra (puede dotársele de una bateria de seguridad).

- **ROM** (Read Only Memory), memoria de sólo lectura, semejante a la RAM, tiene la ventaja de ser permanente (los datos no se borran).

- **Memoria de Línea o de Masa**
  Donde tendremos los ficheros de datos, de acceso por bloque, gran tamaño, lentas y siempre permanentes.

- **Cassette,** de acceso secuencial, cinta magnética, muy lento pero muy barato.
- **Microdrive,** de acceso secuencial, mayor velocidad que el anterior y tamaño medio (85 Kbytes), también cinta magnética.
- **Disco Magnéticos,** flexibles (Floppy Disck) o rígidos (Hard Disck), de acceso aleatorio a bloque, su velocidad es muy aceptable, y de gran tamaño (de 100 Kbytes a 80 Mbytes).

La pila de memoria (Stack Memory) es un sistema de almacenamiento de datos del tipo **LIFO** (Last Input – First Output): Lo último en entrar es lo primero en salir.

Consiste en una pila de datos de 16 bits, funcionando en sentido inverso (crece hacia abajo).

El par **SP** de la CPU contiene la dirección donde se encuentra el último dato almacenado.

Así, si el par SP contiene 50000, el último dato ocupa las posiciones de memoria 50000 y 50001, y el siguiente que entre se colocará en las direcciones 49998 y 49999, decreciendo el valor del par SP a 49998.

En el ZX Spectrum, el sistema coloca el principio del Stack en la dirección señalada por la variable **RAMTOP.** Este valor puede cambiarse por medio de la sentencia **CLEAR** n.

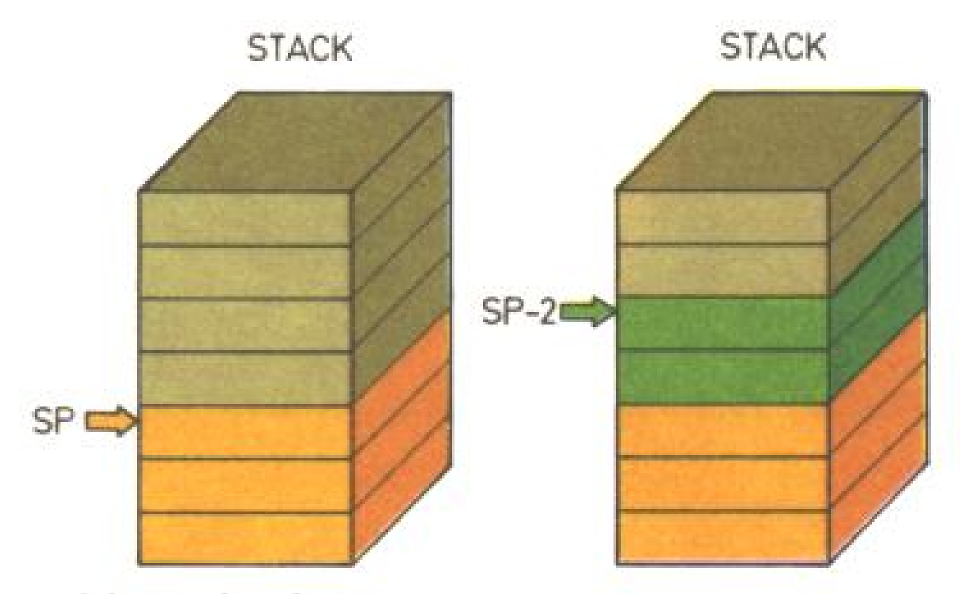
Además de servir para las llamadas (CALL) y retornos (RET) de subrutinas puede utilizarse de los siguientes modos:

Pila	**Utilización**
LIFO	Almacenamiento temporal
Stack Pointer SP	Lista de datos
RAMTOP	Saltos con RET
CLEAR	

- **Almacenamiento temporal de datos:**

  Antes de ejecutar una rutina o un bucle pueden guardarse los registros que se desee preservar mediante la instrucción PUSH y recuperarse después mediante sucesivos POP.

Haciendo:	Se recuperan:
PUSH HL	POP BC
PUSH BC	POP HL



- **Lista de datos:**

  Previamente se sitúa el puntero del STACK señalando al primer dato de la tabla, y posteriormente son leídos los datos mediante sucesivos POP. Una vez finalizada la lectura el puntero (SP) debe recuperar su valor anterior.

- **Saltos diferidos con RET:**

  Si tenemos que guardar una dirección a la que, después de realizar algunas operaciones, tengamos que saltar, podemos escribir, suponiendo que estuviera en el par BC, la secuencia:

```
PUSH BC
operaciones deseadas
RET
```

– **Desbloqueo de la pila**

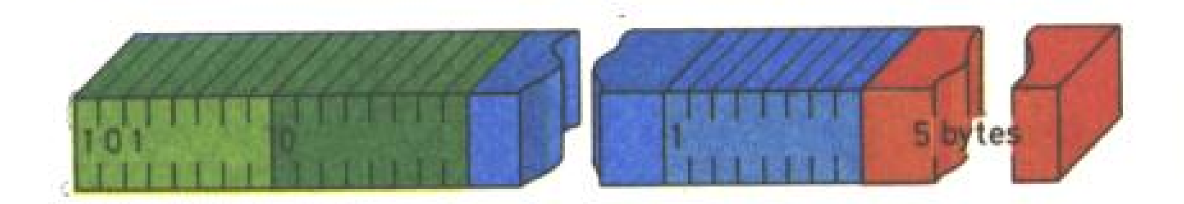
Cuando se detecta error de programación que llena la pila excesivamente, podremos encontrar una dirección de retorno si antes se había guardado el contenido inicial de SP en una parte de la memoria protegida contra este tipo de errores.
Podemos entonces restablecer el contenido del SP, y mediante un RET dirigirnos a un programa de chequeo de errores.

```
LD SP,(ERRSP)
RET
```

Formatos de Variables

Los datos que usamos en BASIC están almacenados en la zona de variables, siguiendo formatos que el intérprete de lenguaje puede identificar, mediante máscaras del código inicial (primer byte). Pueden ser:

Datos de longitud fija	Máscaras
Datos de longitud variable	Valor numérico



Datos de longitud fija:

- **Variable de una sola letra:**
  1 byte. Nombre (máscara 011X XXXX).
  5 bytes con el valor numérico.



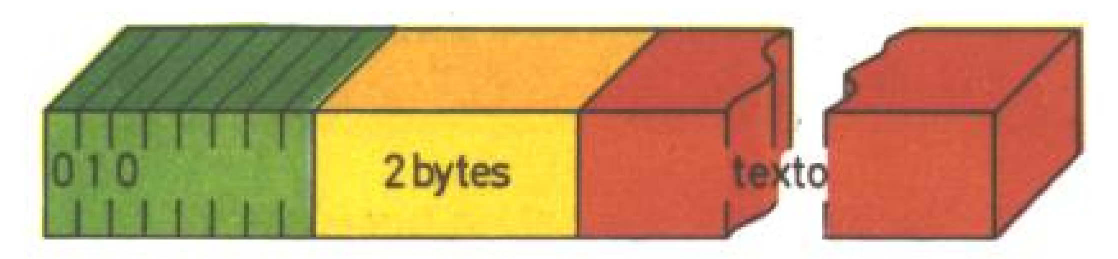
- **Variable de varias letras:**
  1 byte. Primera letra (másc. 101X XXXX).
  n bytes. Siguientes letras (másc. 0XXX XXXX).
  1 byte. Ultima letra (máscara 1XXX XXXX).
  5 bytes con el valor numérico.

- **Variable de control de bucles FOR – NEXT:**
  1 byte. Nombre (máscara 111X XXXX).
  5 bytes para el valor numérico inicial.
  5 bytes para el valor numérico de límite.
  5 bytes valor numérico del paso (STEP).
  2 bytes comienzo del bucle.
  1 byte con el número de sentencia.

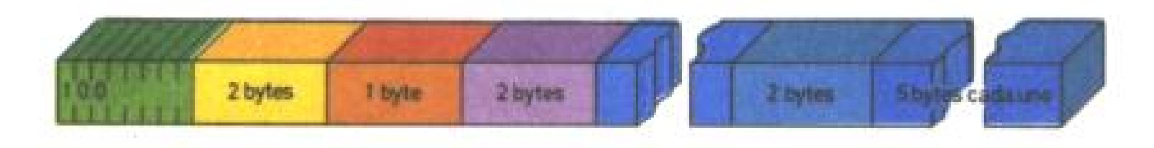


Datos de longitud variable:

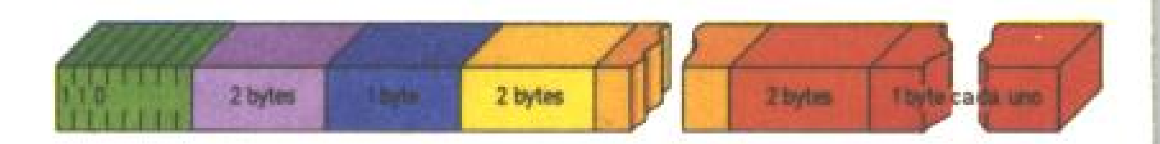
- **Variable de cadena de caracteres:**
  1 byte. Nombre (máscara 010X XXXX).
  2 bytes con la longitud de lo que sigue.
  n bytes para el texto de la cadena.



- **Matriz de elementos numéricos:**
  1 byte. Nombre (máscara 100X XXXX).
  2 bytes con la longitud de lo que sigue.
  1 byte con el número de dimensiones.
  2 bytes por cada dimensión, con el número de elementos de ésta.
  5 bytes para cada elemento.



- **Matriz de caracteres:**
  1 byte. Nombre (máscara 110X XXXX).
  2 bytes con la longitud de lo que sigue.
  1 byte con el número de dimensiones.
  2 bytes por cada dimensión, con el número de caracteres de ésta.
  1 byte para cada carácter de la matriz.

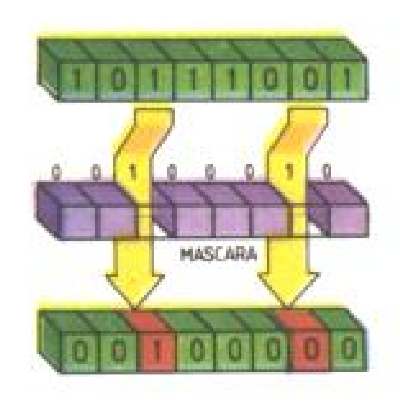


— La **máscara** cubre el código de la letra que identifica la variable.
Así, "A", se transforma en:
Máscara 101X XXXX
Código «A» 0100 0001
Variable A = 1010 0001 = A1H

— Un **valor numérico** (coma flotante) está formado por:
1 byte con el exponente.
4 bytes con la mantisa, siendo su primer bit el signo.

Realiza el producto lógico entre dos bits.
El resultado es 1 si, y sólo si, los dos son 1.
Es 0 si al menos uno de ellos es 0.

El Z80 realiza esta operación con el acumulador y otro registro, posición de memoria o número de 8 bits. El resultado es transferido al acumulador.



• AND A

Mantiene el acumulador con su valor pero ajusta los indicadores, por ello podemos saber:

si A es 0
si es negativo
si hay paridad (número par de unos).

Definición		
AND A		Borrar bits
		Seleccionar bits
	Máscaras	Comprobar bits
		Resto de división
		Contador cíclico

Puede utilizarse también para poner el carry a 0 ya que no existe una instrucción específica que lo haga.

• Máscara AND:

La operación AND puede ser usada para enmascarar los datos. Los 1 de la máscara respetarán el valor inicial, mientras que los 0 ocultarán los valores de los correspondientes bits.

Borrar bits:

La instrucción RES pone a cero un bit en concreto de un byte. La máscara AND puede usarse para sustituir varias instrucciones RES consecutivas.

Seleccionar bits:

Si necesitamos el contenido de parte de un byte, haremos una operación AND entre dicho byte y un dato donde los bits que queremos seleccionar sean 1 y los que queremos borrar sean 0.

De esta manera si queremos aislar los bits 0, 1 y 2 de un byte (por ejemplo para saber la tinta en un byte de atributos), debemos hacer una operación AND con el dato 0000111.

Comprobación de bits:

La máscara deberá llevar 1 en los bits a comprobar y 0 en el resto. Si todos los bits seleccionados son 0 se activará el indicador Z.

Haciendo:

```
LD A,C
AND 00100100B
JP Z,DIR
```

Si los bits 2 y 5 de C son 0, el programa saltará a la dirección DIR, en caso de que al menos uno de ellos fuese 1 el programa seguiría su curso.

Resto de una división:

La función AND n−1 proporciona el resto de la división de A entre n cuando n es potencia de 2.

El número anterior de una potencia de 2 está compuesto por ceros en la parte izquierda y unos en la parte derecha. De esta forma la operación AND permite eliminar la parte más significativa del acumulador.

Contador cíclico:

Si queremos que una variable tome los valores de 0 a x pasando de x nuevamente a 0, siempre que x sea una potencia de 2 menos uno, se enmascara el valor después del incremento con x.

Si realizamos:

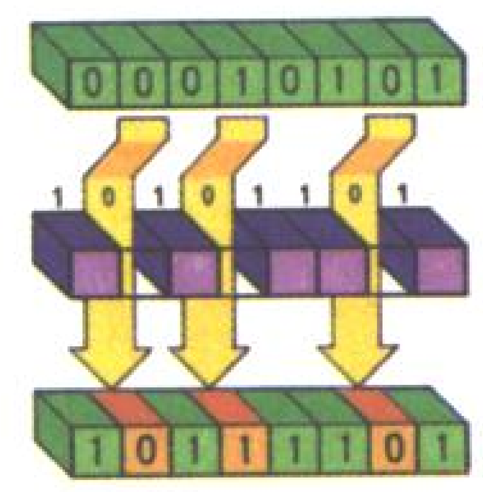
```
LD A,CICL
INC A
AND 00001111B
LD CICL,A
```

Conseguiremos que el valor de la variable CICL cuando llegue a 16 pase a ser 0.

Realiza la suma lógica entre dos bits.

El resultado es 0 si, y sólo si los dos son 0.

El Z80 realiza esta operación con el acumulador y otro registro, posición de memoria, o número de 8 bits. El resultado es transferido al acumulador.



- **OR A:**

Mantiene el acumulador con su valor pero ajusta los indicadores, por ello podemos saber:

si A es 0
si es negativo
si hay paridad (número par de 1s)

Definición OR A		
		Asignar bits
	Máscáras	Añadir bits
		Comprobar bits
		Comprobar palabra

Puede utilizarse también para poner el carry a 0 ya que no existe una instrucción específica que lo realice.

- **Máscara OR:**

La operación OR puede ser usada para enmascarar los datos. Los 0 de la máscara respeterán el valor inicial, mientras que los 1 ocultará los valores de los correspondientes bits.

Asignar bits:

La instrucción SET pone a 1 un bit concreto de un byte. La máscara OR puede usarse para sustituir varias instrucciones SET consecutivas.

Componer byte:

La operación OR puede usarse para reponer la parte de un byte eliminada por AND.

Supongamos que queremos sutituir los 3 bits bajos del registro B por los del registro C:

```
LD    A,B
AND   11111000B    ; Borra de B los tres
LD    B,A          ; bits bajos.

LD    A,C          ; Sitúa en A los tres
AND   00000111B    ; bits bajos de C.

OR    B            ; Une las dos partes.
LD    B,A          ; Lo carga en B.
```

Comprobación de bits:

Se utiliza para comprobar si una serie de bits son 1.

La máscara deberá llevar 0 en los bits por comprobar y 1 en el resto. Si todos los bits seleccionados son 1 al incrementar el resultado dará 0, por lo que se activará el indicador Z.

Haciendo:

```
LD    A,C
OR    11011011B
INC   A
JP    Z,DIR
```

Si los bits 2 y 5 de C son 1, el programa saltará a la dirección DIR, en caso de que al menos uno de ellos fuese 0 el programa seguirá su curso.

Comprobación de palabra:

Para comprobar si el valor de los bytes que componen una palabra es 0 se carga uno de ellos en el acumulador y se hace OR con el resto.

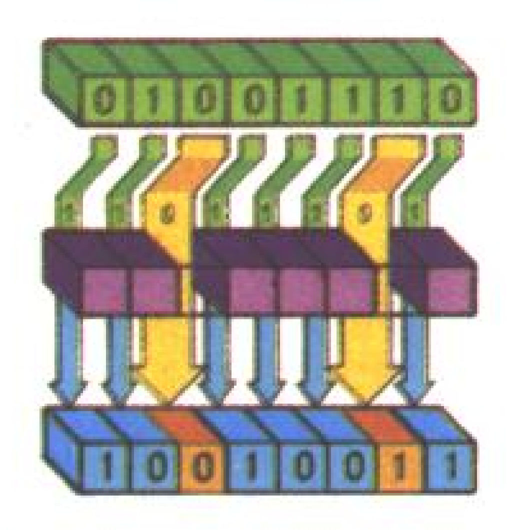
```
LD    A,B
OR    C
JP    NZ,DIR
```

En caso de que tanto B como C sean 0 la rutina seguirá su curso. Si alguno de ambos no fuese 0 saltaría a la dirección DIR.

Realiza la comparación lógica entre dos bytes, bit a bit.

El resultado es 1 si son diferentes.

Es 0 si los dos son iguales.



- **XOR A:**

Normalmente se usa para poner el acumulador a 0, salvo cuando quieran respetarse los flags, en cuyo caso deberá hacerse LD A,0.

Los indicadores Z y P/V (indicador de paridad) son puestos a 1 y el resto a 0, por lo que F resulta con el valor 68, (44H).

Definición		
XOR A		Complementar bits
		Comp. el acumulador
	Máscaras	Comparar bits
		Suma sin carry
		Cifrado
		Pintar en OVER 1

- **Máscara XOR:**

Los 0 de la máscara XOR respetan el valor inicial al igual que OR, pero los 1 tienen la particularidad de complementar el valor:

Los unos pasan a ser ceros y los ceros unos.

Es debido a esto por lo que máscara XOR posee la característica de la reversibilidad. Una segunda máscara equivalente devuelve el valor inicial.

Complementar bits:

Con el siguiente ejemplo complementamos los bits 3 y 5 del byte BAND:

```
        LD      A, (BAND)
        XOR     00101000B
        LD      (BAND),A
```

Complementación del acumulador (byte):

Al igual que la instrucción CPL la operación XOR 11111111B (FFH) complementa todo el byte del acumulador pero con la diferencia de que afecta a todos los indicadores, mientras CPL no.

Comparación de bits:

```
        LD      A,B
        XOR     C
        BIT     3,A
        JR      Z,EQU
```

En el caso de que el bit 3 de B y el bit 3 de C sean iguales el programa saltará a la rutina EQU, si son distintos seguirá su curso.

Suma sin carry:

La operación XOR efectúa la llamada suma sin carry o suma NIM, que consiste en sumar sin tener en cuenta el acarreo de un bit al siguiente. Puede ser útil en análisis de juegos, control de paridad, etc.

Cifrado de textos y programas:

La reversibilidad de la máscara XOR hace posible su utilización como clave, existiendo pues, 255 claves diferentes.

```
        LD      BC, longitud
        LD      HL, comienzo
BUCLE   LD      A, (HL)
        XOR     clave
        LD      (HL) ,A
        DEC     BC
        LD      A, B
        OR      C
        JR      NZ, BUCLE
```

Esta rutina sirve tanto para cifrar como para descifrar un bloque de bytes.

Pintar en OVER 1:

Este modo de dibujo consiste en superponer dos figuras con la operación XOR.

Constantes:

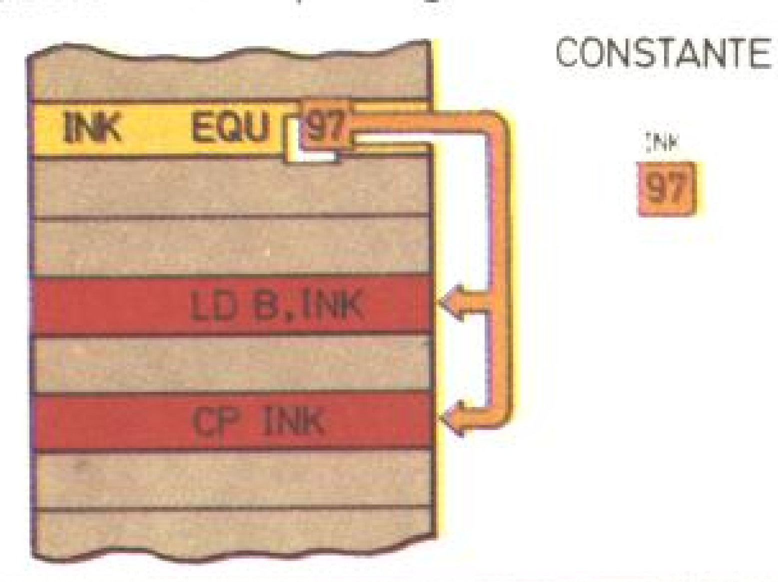
Son valores numéricos que permanecen inalterables a lo largo del programa.

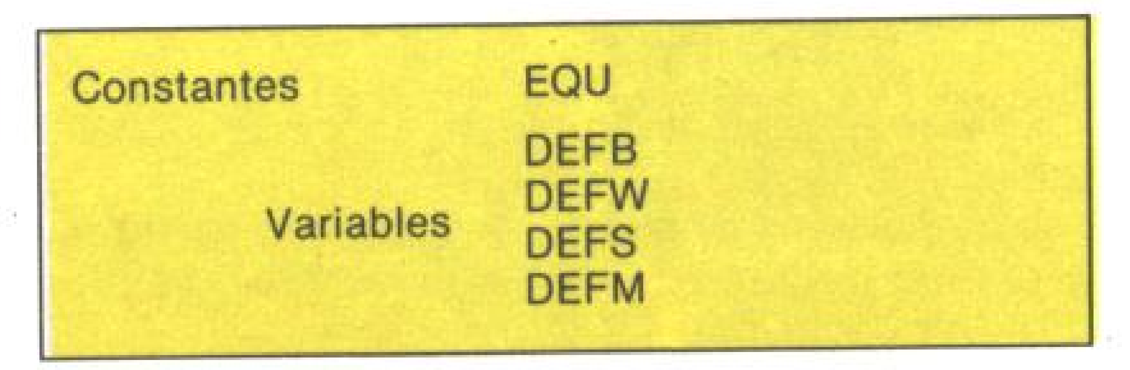
Puede ser útil declararlas con etiquetas por las siguientes razones:

— Mayor claridad en el programa.

— Sustituir ese valor de una sola vez en todos los lugares donde aparece, en caso de modificación del programa.

Las constantes se declaran con la seudoinstrucción **EQU** que significa «equivale».





Ejemplo:

INK EQU 97

Significa que en todos los lugares donde aparece la etiqueta EQU debe ponerse el número 97

Variables:

Cuando los registros no son suficientes para almacenar un valor, se habilita un lugar en la memoria.

Para determinar ese lugar puede utilizarse en el lenguaje ensamblador la dirección en que se encuentra, definiéndola mediante **EQU:**

INK EQU 53000

53000 es la dirección donde se situará la variable.

A menudo es conveniente situar la variable en el interior del código objeto; para ello se utilizan los seudomnemónicos siguientes:

DEFB para un byte o una serie de bytes separados por comas (puede ser un número o un caracter entrecomillado).

DEFW para una palabra (dos bytes) o una serie de palabras, separadas por comas.

DEFS deja un espacio de un número de bytes a los que no asigna ningún valor inicial.

DEFM crea un espacio conteniendo un texto, que debe ir entre comillas.

Para manejar variables debemos ponerla entre paréntesis que significa «el contenido de».

Ejemplo:

Inicializamos un byte a cero y lo almacenamos en una dirección que llamaremos INK con la instrucción:

```
INK DEFB 0
```

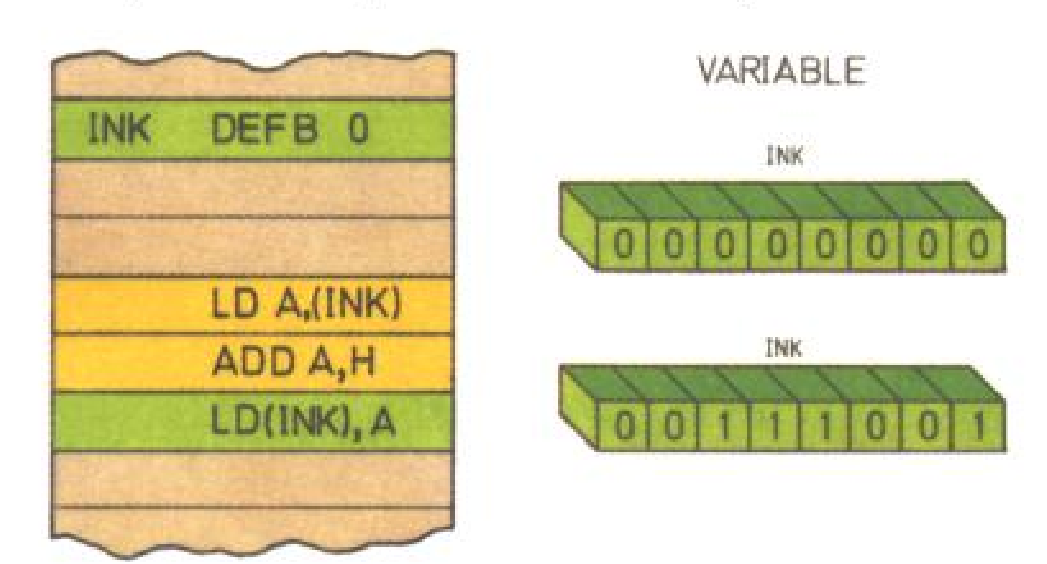
Cargamos en el acumulador A el byte situado en la dirección INK:

```
LD      A, (INK)
```

Sumamos al acumulador A el registro H que tiene el número 57 en binario, finalmente cargamos en INK el valor del acumulador A:

```
ADD   A,H
LD    (INK),A
```

A partir de ahora INK tendrá el mismo contenido que H + A (en este caso 57).



Los indicadores o banderas consisten en una información de un solo bit. Sólo pueden tener dos valores 1 ó 0, que se identifican con sí o no.

Esta información es muy útil a la hora de la toma de decisiones en un programa ante una bifurcación.

- Las instrucciones relacionadas con las banderas son SET, RES y BIT:

SET alza una bandera (indicador 1).

RES baja una bandera (indicador 0).

BIT comprueba el estado de un indicador y, conforme a ello, sitúa su bandera interna Z del registro F. (Z si es 0; NZ si es 1).

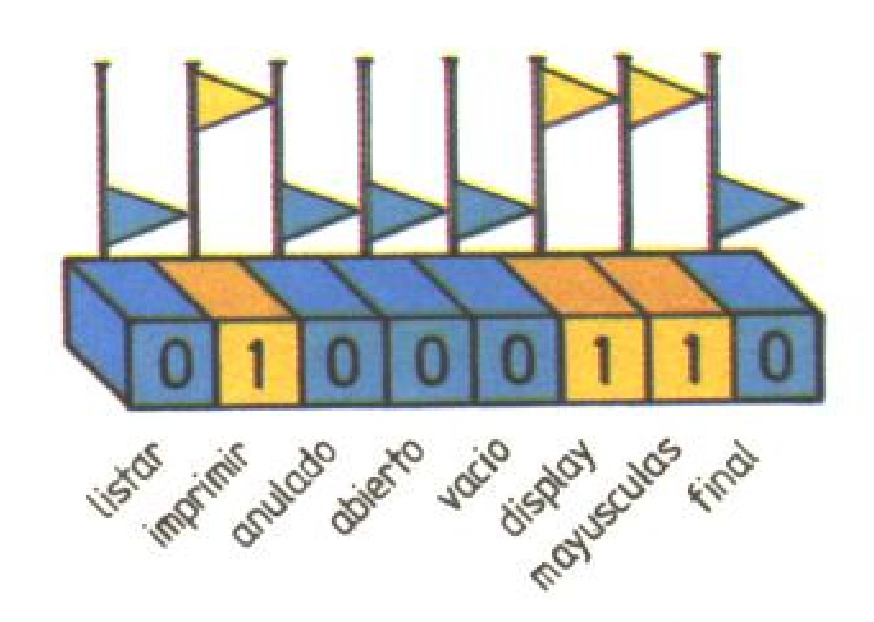
Banderas del microprocesador:

Son los indicadores del registro F, ya explicadas en la correspondiente ficha.

Banderas del sistema:

El intérprete Basic utiliza una serie de VARIABLES DEL SISTEMA, algunas de las cuales son utilizadas en forma de banderas (información bit a bit).

Definición	Banderas del micro
Utilización	Banderas del sistema
Instrucciones relacionadas	Banderas del programa
	Cambio de estado



Estas se consultan continuamente para determinar cuáles son las rutinas que deben ejecutarse en cada momento.

Banderas de programa

En cualquier programa pueden usarse banderas de un modo similar al del intérprete BASIC.

Para ello debe asignársele un espacio en una determinada zona de memoria directamente mediante EQU o, reservarse con el propio ensamblador mediante un seudomnemónico DEF. (ver ficha variables).

De esta forma:

```
BAND DEFB 0
```

Establece un espacio para un byte llamado BAND y lo inicializa con todos sus bits a 0.

```
LD    HL,BAND
SET   3,(HL)
```

pone a 1 el bit 3 del byte BAND.

```
LD    HL,BAND
BIT   3,(HL)
JP    Z,DIR1
```

salta a la dirección DIR1 en caso de que esté alzada la bandera del bit 3, en caso contrario continúa por su curso normal.

Cambio del estado de una bandera

En algún momento puede necesitarse invertir el valor de una bandera; ponerla a 0 si está a 1 y a 1 si está a 0 sin conocer previamente su valor. Esto puede hacerse mediante una instrucción XOR:

```
LD    A ,(BAND)
XOR   00001000B
LD    (BAND) ,A
```

De esta forma invertimos el valor del bit 3 del byte BAND.

Indicadores del sistema

Las variables del sistema siguientes son las que contienen los indicadores o banderas que utiliza el intérprete BASIC:

FLAGS	FLAGX
TV FLAG	P FLAG
FLAGS2	FLAGS3

— FLAGS (23611), (IY + 1), (5C3BH)

Contiene varias banderas que controlan el BASIC.

Bit 0: No se pone ningún espacio ante del próximo comando.

Bit 1: Impresión en pantalla (1) o impresora (0).

Bit 2: Se utiliza el modo K.
Es 1 si se está utilizando el modo L.

Bit 3: Modo L en un INPUT.

Bit 5: Indica que una tecla se ha pulsado en conjunción con LASTK.

Bit 6: La expresión es numérica (1) o de caracteres (0).

Bit 7: Se está ejecutando una orden.
Es 0 cuando el intérprete BASIC está chequeando la sintáxis de una línea.

— TV FLAG (23612), (IY + 2), (5C3CH)

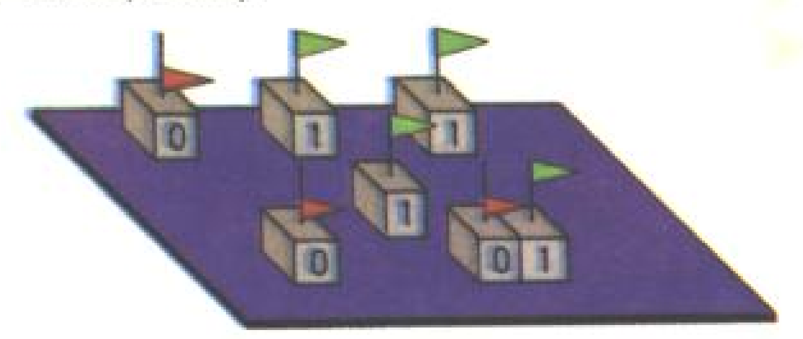
Indicadores relacionados con la televisión.

Bit 0: Se está trabajando en la parte inferior de la pantalla.

Bit 3: El modo ha cambiado y debe ser chequeado otra vez.

Bit 4: Se está en un listado automático.

Bit 5: La parte inferior de la pantalla ha de ser limpiada para situar una información (un código de error, etc.).



— **FLAGS2 (23658), (IY + 48), (5C6AH)**

Bit 0: Es innecesario que la pantalla se limpie cuando una línea es introducida dentro del área de edición.
Bit 1: El buffer de impresora ha sido utilizado por la ROM de 16 K.
Bit 2: La pantalla está limpia.
Bit 3: Se está en mayúsculas.
Bit 4: Se está utilizando el canal K.

— **FLAGX (23665), (IY + 55), (5C71H)**

Bit 0: La expresión tratada es una cadena simple.
Bit 1: Se está asignando una nueva variable.
Bit 5: Se está ejecutando una sentencia INPUT.
Bit 6: El INPUT es alfanumérico.
Bit 7: Se está ejecutando un INPUT LINE.

— **P FLAG (23697), (IY + 87), (5C91H)**

Se utiliza para discriminar los parámetros del PRINT. Los bits impares se refieren a los parámetros permanentes, y los pares a los temporales.
Bits 1 y 2: OVER.
Bits 2 y 3: INVERSE.
Bits 4 y 5: INK 9.
Bits 6 y 7: PAPER 9.

— **FLAGS 3 (23734), (IY + 124), (5CB6H)**

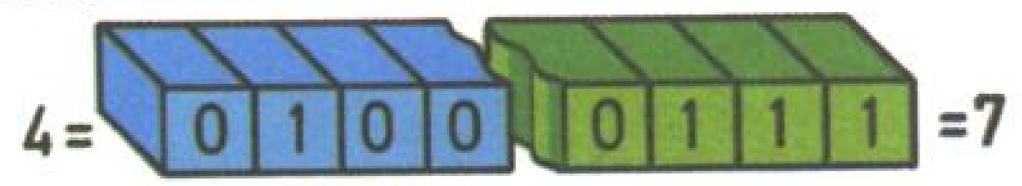
Este byte de indicadores pertenece a las nuevas variables que utiliza la ROM de 8 K del INTERFACE 1.
Bit 0: Se está ejecutando un comando extendido.
Bit 1: Se ejecuta CLEAR#.
Bit 2: ERR SP ha sido alterado por la ROM del interface 1.
Bit 3: Está ejecutándose una rutina que afecta a la red local.
Bit 4: Ejecutando LOAD *
Bit 5: Ejecutando SAVE *
Bit 6: Ejecutando MERGE *
Bit 7: Ejecutando VERIFY *

Cuando cada dígito de una cantidad se representa por un conjunto de 4 bits, se dice que dicha cantidad está codificada en **BCD** («Decimal Codificado en Binario»).

Así, por ejemplo, el byte 01000111B que corresponde en codificación ordinaria con 71 decimal, codificado en BCD correspondería al número decimal 47 (0100 = 4 y 0111 = 7).

Para esto, sólo necesitamos los 10 primeros números de los 16 posibles con 4 bits, esto es, usamos los valores del 0 al 9 y no se utilizan de la A a la F.

El valor decimal de un número en BCD coincide con la grafía de la notación hexadecimal del valor del byte. Así 27H = 27 , 88H = 88. Por otra parte, F4H o 1AH no tendrían sentido en BCD.



Decimal codificado en Binario
- Representación
- Utilización

DAA

RLD y RRD
- Rutina de impresión

La utilización de números BCD tiene el inconveniente de su dificultad de manejo pero, por otra parte, simplifica considerablemente la representación gráfica. Son pues aconsejables en los casos en que se necesitan pocos cálculos y sencillos, y representación gráfica rápida. (Ej: marcador de puntuación de un juego).

• DAA

Cuando el ordenador suma o resta números codificados en BCD, realiza la operación en forma binaria siendo el resultado muchas veces erróneo en BCD, por exceder las cifras del valor 9.

La instrucción DAA modifica estos resultados realizando una suma de compensación de 00H, 06H, 60H ó 66H según el caso.

Para funcionar correctamente, la instrucción DDA necesita los flags H y N, por lo que no se deben intercalar instrucciones que afecten a los flags entre una operación aritmética y DAA.

Ejemplo:

```
LD   A,73H
LD   B,18H
ADD  A,B   ; A vale 8BH sin sentido en BCD
DAA        ; A vale 91H = 91 BCD
```

● **RLD y RRD**

Estas instrucciones producen una rotación de dígito a izquierda o derecha entre el acumulador y el contenido de la dirección señalada por HL [(HL)].

Son muy útiles en el manejo de números en BCD.

Ejemplo:

```
        LD    B,NBY    ;Numero de bytes
        LD    HL,DIR   ;Direcc. primer byte
BUCLE   LD    A,"0"    ;0 ascii en el ac.
        RLD   ;        ;Primer digito
        PUSH  AF       ;Guarda acumulador
        RST   16       ;Lo imprime
        POP   AF       ;Recupera acumulador
        RLD   ;        ;Segundo digito
        PUSH  AF       ;Guarda acumulador
        RST   16       ;Lo imprime
        POP   AF       ;Recupera acumulador
        RLD   ;        ;Restablece el byte
        INC   HL       ;Siguiente byte
        DJNZ  BUCLE    ;Continua bucle.
```

Esta rutina muestra la forma de imprimir un número BCD de cualquier longitud.

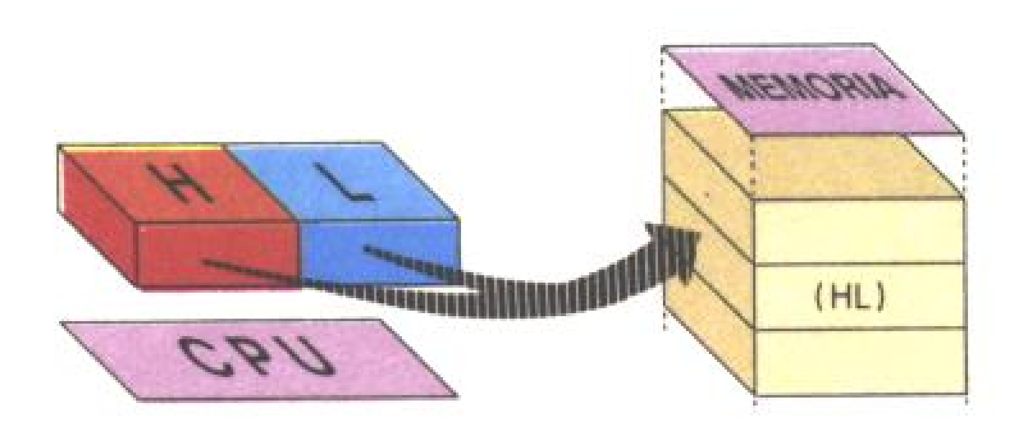
Puntero es todo registro o posición de memoria que contiene la dirección de cualquier dato, texto, dibujo, etc. Se dice que «señala» a esa dirección.

Así, por ejemplo, las direcciones de memoria 23635 y 23636 (Variable del sistema PROG) señalan el comienzo del BASIC.

Definición	
Direccionamiento por:	Registro Constante Variable Indices
Tablas simples y dimensionadas	

- **Registros puntero:**

Los punteros PC y SP señalan respectivamente la dirección del programa que se está ejecutando y la dirección de la pila o stack.



Los registros índice y el par de registros HL están pensados especialmente para hacer de puntero. (Existen una serie de instrucciones que afectan especialmente al contenido de la dirección señalada por HL, IX + d o IY + d). Pero, aunque con algunas restricciones, también pueden servir de puntero los pares de registros DE y BC.

- **Números puntero (Constantes):**

Para obtener un dato de una dirección señalada por una constante basta con leerlo en la forma:

```
LD    A,(DIR)
```

si es de un byte, o:

```
LD    HL,(DIR)
```

si es de dos bytes.

- **Variables puntero:**

Para leer un dato señalado por una variable, en primer lugar deberemos obtener el valor de esa variable y después el dato deseado:

Para un byte:	Para dos bytes:
	LD HL,(VAR)
LD HL,(VAR)	LD E,(HL)
LD A,(HL)	INC HL
	LD D,(HL)

- **Indices:**

IX e IY son unos punteros especiales, pues direccionan la base de una tabla de 256 posibles datos mediante el modo de direccionamiento indexado.

- **Tablas de datos:**

Si tenemos una serie de datos señalados por una variable podremos acceder a todos ellos directamente asignando a uno de los registros índice el valor de esa variable. Así mediante:

```
LD    IX,(TABLA)
LD    A,(!X + 8)
```

tendremos en A el octavo dato de la tabla.

- **Tablas dimensionadas:**

Supongamos que tenemos una tabla de 4 grupos de 3 datos y que la base de la misma está señalada por el par de registros IX y queremos obtener el segundo dato del tercer grupo, deberemos hacer:

```
LD    DE,3         ; Longitud de los grupos
LD    HL,2         ; Número de grupo menos 1
CALL  30A9H        ; HL = HL*DE (ROM).
EX    DE,HL        ; Intercambia DE y HL
ADD   IX,DE        ; Suma a IX la longitud de
                   ; los grupos anteriores
LD    A,(IX + 1)   ; 2.º dato del 3.er grupo
```

El comienzo del BASIC viene determinado por la variable PROG (23655).

Línea Basic:

Cada línea BASIC consta de:

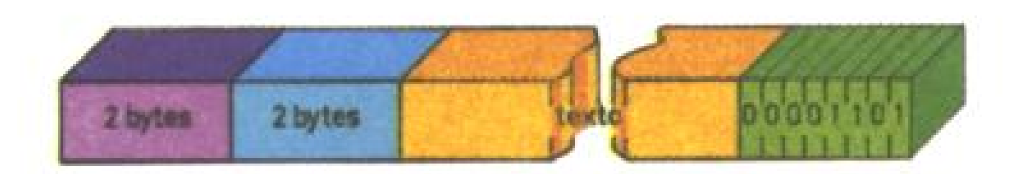
2 bytes de número de línea colocados a la inversa de la forma habitual para la CPU, pues del primero es el alto y el segundo es el bajo.

2 bytes con la longitud de lo siguiente (de la forma habitual: primero el byte bajo y después el alto).

N bytes que forman el cuerpo de la línea.

1 byte de fin de línea que siempre es el caracter ASCII 13 (Retorno de carro).

• En el interior de la línea BASIC existen las siguiente particularidades:



Línea BASIC	
Tokens	DEF FN
Números	DATA

Tokens:

Son las palabras-clave o comandos BASIC, que ocupan un solo byte, aunque la representación en pantalla sea de varios caracteres.

Números:

Constan de dos partes:

— La representación ASCII el mismo, que sirve para la representación en el listado.

— El número codificado en coma flotante, que no se ve en el listado y que es el que usa el ordenador. Esta codificación usa 6 bytes:

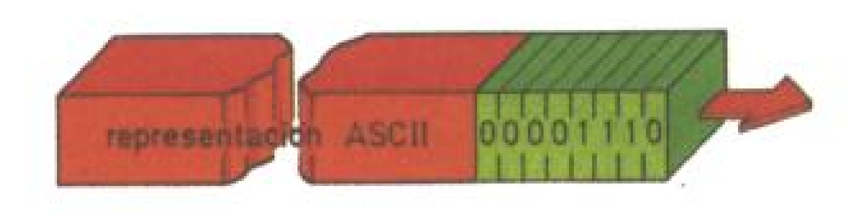
1 byte código 14 de identificación , que indica que a continuación hay un número codificado en coma flotante.

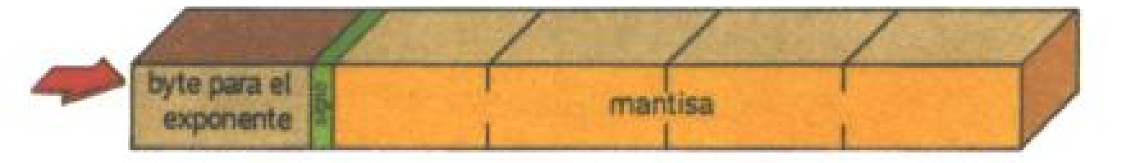
5 bytes para la representación:

1 byte de exponente.
1 bit de signo.
31 bits (4 bytes— 1 bit) de mantisa.

Los números enteros menores de 65535 ocupan los bytes penúltimo y antepenúltimo.

Por ello cada número ocupa una memoria igual al número de sus cifras +6 bytes.





DEF FN:

En una sentencia tipo DEF FN F (A,B$,C) = N cada uno de los parámetros entre paréntesis reserva un espacio de 5 bytes , separado por un caracter código 14 al igual que los números.

En principio contiene valores indeterminados. Al ejecutarse la función (FN) son cubiertos de la siguiente forma:

— Parámetros numéricos: se guarda el valor en coma flotante de la forma habitual.

— Parámetros alfanuméricos:

1 byte de tipo: 0 variable dimensionada,
1 variable sin dimensionar,
44 texto.

2 bytes que indican la dirección donde se encuentra el texto.
2 bytes con la longitud del mismo.

Setencias DATA:

Los datos se encuentran de forma similar a como en el resto del Basic: los datos alfanuméricos se almacenan tal como se ve en pantalla y los numéricos tienen 5 bytes ocultos tras el caracter código 14.

De esta forma < < 15 > > ocupará 8 bytes mientras que < < "15" > > solamente 4.

Mapa de memoria

Los 64 KBytes (0000-FFFFH,0-65535d) de memoria están distribuidos en zonas que pueden ser de 4 tipos diferentes:

● **Zonas fijas:**

Son las que se encuentran en la parte más baja, y siempre ocupan el mismo espacio. Son:

— La **ROM**. (0-3FFFH,0-16383,16KB). Es la memoria permanente de «sólo lectura» que contiene los programas de sistema operativo y editor e intérprete de Basic, así como el juego de caracteres.

— El **«display file»** o fichero de pantalla (4000H-57FFH,16384-22527,6KB), donde se encuentran los pixels o puntos que forman los gráficos y los caracteres.

— El **«attribute file»** o fichero de atributos (5800H-5AFFH,22528-23295,768), donde se hallan los códigos de los atributos de color.

— **«Buffer de impresora»** (5B00H-5B00H, 23296d-23551d,256): Almacenan temporalmente los caracteres hasta completar una línea.

Zonas fijas	Sistema operativo (ROM). Display file. Attribute file. Variables del sistema.
Zonas dinámicas	Bajas. Espacio de separación. Altas.
Zonas libres	

— Las **variables del sistema** (5C00H-5CBCH,23552-23733,182), que contienen información precisa para los programas de la ROM.

● **Zonas dinámicas bajas:**

Son las que se sitúan a continuación de las anteriores, pueden desplazarse o crecer hacia arriba según las necesidades de la ROM:

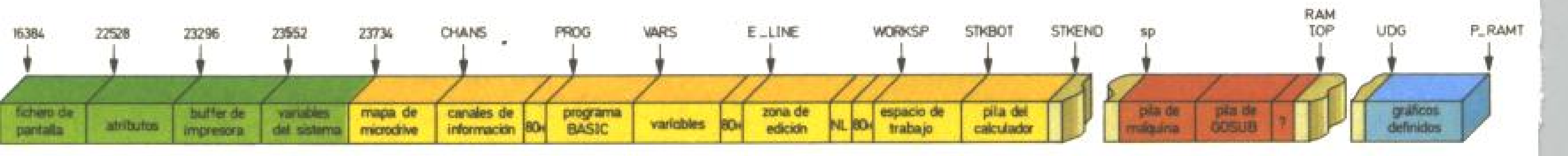
— **Ampliación de variables del sistema** (57) y mapas de microdrive (cada mapa ocupa 32B y vale para un drive), que se colocan sólo cuando el interface 1 está conectado.

— **Información de canales** con una longitud mínima de 20 bytes (5 por cada canal K,S,P o R), si se conecta el interface 1 cada canal M ocupa 595 bytes, cada canal N 276 y cada canal B o T 11.

— **Programa Basic,** cuya longitud será la suma de todas las longitudes de las líneas que lo forman.

● **Zonas dinámicas altas:**

A partir de las zonas dinámicas bajas normalmente queda un espacio libre para ampliar el Basic hasta llegar a la pila de máquina, que se encuentra inmediatamente anterior a la dirección indicada por la variable de sistema RAMTOP (5CB2H,23730d), y que contiene las direcciones de retorno en código máquina o Basic.



— **Variables del programa Basic** de longitud dependiente de las variables que éste utilice.

— **Area de edición,** donde se sitúa una línea editada.

— **Espacio de trabajo** área auxiliar, que utiliza el calculador en operaciones con cadenas de caracteres.

— **Pila del calculador** que el calculador utiliza en las operaciones en coma flotante.

● **Zonas libres:**

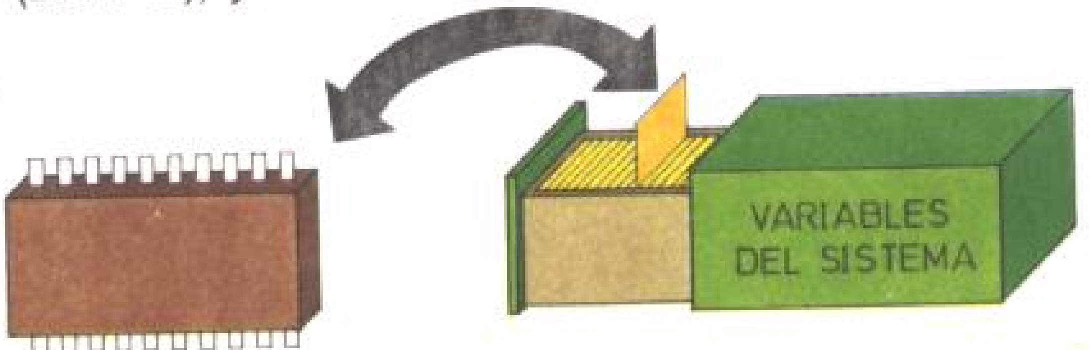
Por encima de RAMTOP queda un espacio libre para el usuario hasta la dirección indicada por la variable de sistema PRAMT (5CB4H,23732d) o el final de la memoria, del que la ROM sólo utiliza la zona de **gráficos definibles** que comienza en la dirección indicada por la variable de sistema UDG (5C7BH,23675d) y de 168 bytes de longitud.

Las variables del sistema son utilizadas por el sistema operativo del ordenador para señalar las diferentes partes en que está distribuida la memoria, para decidir qué rutinas utilizar según los canales que se estén usando.

En suma, para guardar todos aquellos datos de interés y que no tienen cabida en los registros internos del microprocesador.

Lo más interesante es que estas variables, al estar en RAM no sólo se pueden consultar, sino que pueden ser modificadas según las necesidades o exigencias de nuestros programas.

Las variables del sistema se almacenan desde la dirección 5C00H (23552d) hasta la CBCH (23734d), y son:



STRLEN	BORDCR	ATTR-T
SEDD	ATTR-P	MASK-T
FRAMES	MASK-P	P-FLAG

— **STRLEN IY + 56 5C72H 2366d 2 bytes**

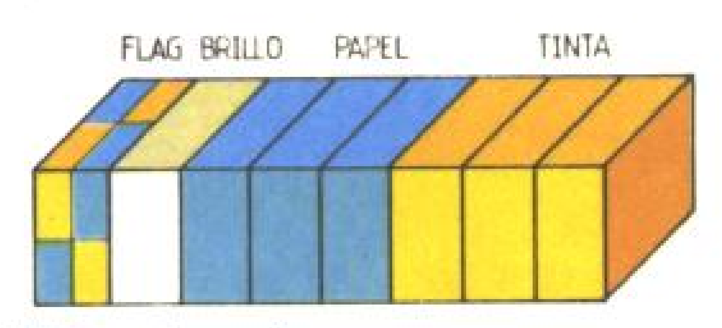
Contiene, si se está usando una variable alfanumérica, su longitud. Si la variable es numérica o una nueva alfanumérica, contiene en su byte bajo, el código de la letra que identifica la variable. Es usada por FOR (1D03H) y LET (2AEEH).

— **SEED IY + 60 5C76H 23670d 2 bytes**

Base de la serie de números aleatorios (función RND). Es asignada por la función RANDOMIZE (1E4FH).

— **FRAMES IY + 62 5C78H 23672d 3 bytes**

Contador incrementado 50 veces por segundo por la rutina RST 38, de las interrupciones enmascarables. Es usada por la función RANDOMIZE (1E4FH) para copiar su valor si no le es asignado ninguno.



- **Variables de color:**

— BORDCR IY + 14 5C48H 23624d 1 byte

Contiene el color de la parte inferior de la pantalla y el del borde. Haciendo POKE puede conseguirse asignar FLASH, BRILLO y TINTA.

— ATTR-P IY + 83 5C8DH 23693d 1byte

Contiene los colores permanentes. Es asignada por las instrucciones PAPER, INK, BRIGHT y FLASH.
Es utilizada por la rutina TEMPS (0D4DH) para copiar el valor en ATTR-T.

— MASK-P IY + 84 5C8EH 23694d 1 byte

Máscara para colores transparentes permanentes (color 8). Los bits a 1 indican que el color no debe tomarse de ATTRP, sino mantener los que haya en pantalla. Es utilizada por TEMPS para copiar su valor en MASK-T.

— ATTR-T IY + 85 5C8FH 23695d 1 byte

Número de color temporal asignado en el interior de sentencias PRINT, DRAW, etc. En caso contrario se mantiene el de ATTR-P copiado por la rutina TEMPS (0D4DH). En todo caso, las instrucciones de presentación en pantalla utilizan esta variable y MASK-T.

— MASK-T IY + 86 5C90H 23696d 1byte

Como MASK-T, pero para los colores temporales. Es usada en conjunción con ATTR-T y P-FLAG para asignar un atributo por la rutina PO-ATTR (0BDBH).

— P-FLAG IY + 87 5C91H 23697d 1 byte

Utilizada para los parámetros OVER, INVERSE e INK 9. Ver microficha G-23.

Existen una serie de variables del sistema que señalan las posiciones donde ha de colocarse el siguiente carácter que deba presentarse:

DF-SC	SPOSN
COORDS	SPOSNL
ECHO-E	SCR-CT
DF-CC	P-POSN
DF-CCL	PR-CC

- **Punteros de pantalla:**

— DF SZ IY + 49 5C6BH 23659d 1byte

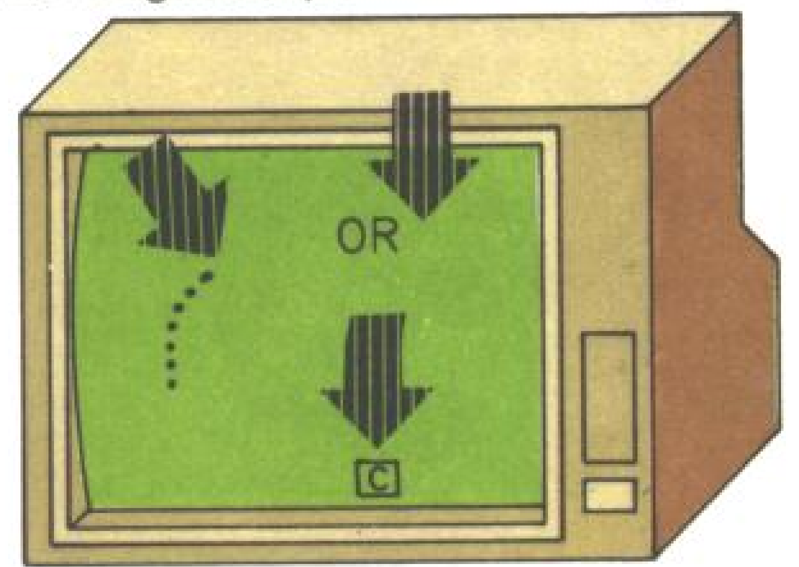
Contiene el número de líneas que hay en la parte inferior de la pantalla.

— COORDS IY + 67 5C7DH 23677H 2 by.

Coordenadas del último punto dibujado en pantalla por alguna de las instrucciones PLOT (22DCH), DRAW (2382H) o CIRCLE (2320H). Es puesta a 0 por CL-ALL (0DAFH) en la ejecución de las sentencias NEW, CLEAR y CLS. Se utiliza como punto de partida para una próxima instrucción DRAW.

— ECHO-E IY + 72 5C82H 23682d 2 bytes

Contiene 33, menos el número de columna; y 24, menos el número de línea de la próxima posición de PRINT, en la parte inferior de la pantalla. Es asignada por PO-STORE (0ADCH).



— **DF-CC IY+74 5C84H 23684H 2 bytes**

Contiene la dirección del pixel superior izquierdo de la siguiente posición de PRINT. Es asignada por PO-STORE (0ADCH).

— **DF-CCL IY+76 5C86H 23686d 2 bytes**

Igual que DF-CC, pero para la parte inferior de la pantalla.

— **S-POSN IY+78 5C88H 23688d 2 bytes**

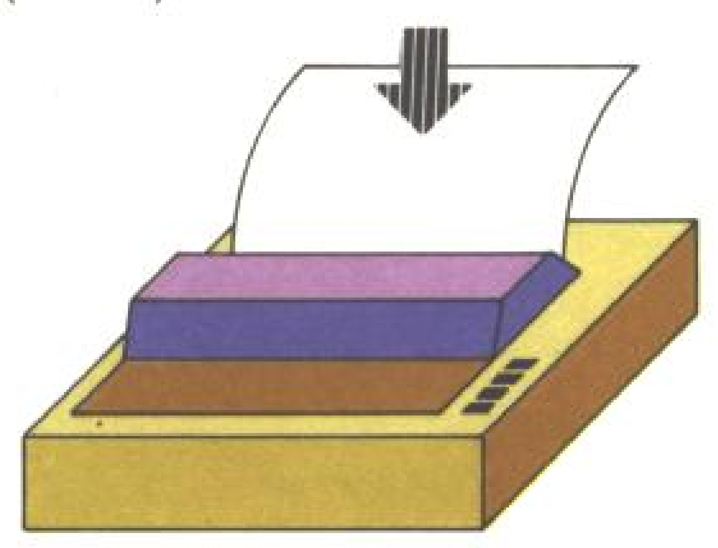
Contiene 33, menos el número de columna; y 24 menos el número de línea de la próxima posición de PRINT en la parte superior de la pantalla. Es asignada por PO-STORE (0ADCH).

— **SPOSNL IY+80 5C8AH 23690d 2 by.**

Lo mismo que ECHO-E. Esta variable está duplicada por necesidades del EDITOR.

— **SCR-CT IY+82 5C8CH 23692d 2 by.**

Contador de Scroll. Contiene el número de veces que ha de desplazarse el texto antes de que aparezca el mensaje «Scroll?». Es utilizada por las rutinas PO-SCR (0C55H), CL-ALL (ODAFH) e INPUT (2089H).



● **Punteros de impresora:**

— **P-POSN IY+69 5C7FH 23679d 1 byte**

Contiene 33, menos el número de columna en el buffer de impresora.

— **PR-CC IY+70 5C80H 23680d 1 byte**

Byte menos significativo de la dirección que señala P-POSN.

Este conjunto de catorce variables del sistema consisten en una serie de punteros que señalan las diferentes secciones del programa así como otros datos de interés.

Toda la zona del Basic es susceptible de cambiar de lugar. Cada vez que se añade o se elimina un byte en uno de sus puntos, los punteros son actualizados por la rutina POINTERS (1664H).

VARS	NXTLIN	X-PTR
DEST	DATADD	WORK-SP
CHANS	E-LINE	STKBOT
CURCHL	K-CUR	STKEND
PROG	CH-ADD	

— VARS IY + 17 5C4BH 23627d 2 bytes

Contiene la dirección donde comienzan las variables Basic.

— DEST IY + 19 5C4DH 23629d 2 bytes

Contiene la dirección de la variable que está asignándose. Puede utilizarse en una rutina código máquina llamada de forma:

Let N = USR...

— CHANS IY + 21 5C4FH 23631d 2 bytes

Almacena la dirección del comienzo del área de los canales de información.

— CURCHL IY + 23 5C51H 23633d 2 by.

Contiene la dirección del comienzo de la información del área de los canales de información para el canal en uso.

— PROG IY-25 5C53H 23655d 2 bytes

Contiene la dirección de inicio del área de programa Basic.

— NXTLIN IY + 27 5C55H 23637d 2 by.

Contiene la dirección de la siguiente línea de programa.

Puede usarse para intercambiar datos con el código máquina en la línea siguiente a la que se encuentre la llamada USR.

— DATADD IY + 29 5C57H 23639d 2 by.

Contiene la dirección de la última coma utilizada en una sentencia DATA, o el comienzo de

una línea dada por un RESTORE, o la siguiente si no existe.

— E LINE IY + 31 5C59H 23641d 2 bytes

Contiene la dirección del área de edición que está detrás de las variables. Es usada por el EDITOR (0F2CH).

— K CUR IY + 33 5C5BH 23643d 2 bytes

Contiene la dirección del cursor en la línea que se está editando. Usada por ADD-CHAR (0F81H).

— CH ADD IY + 35 5C5DH 23645d 2 by.

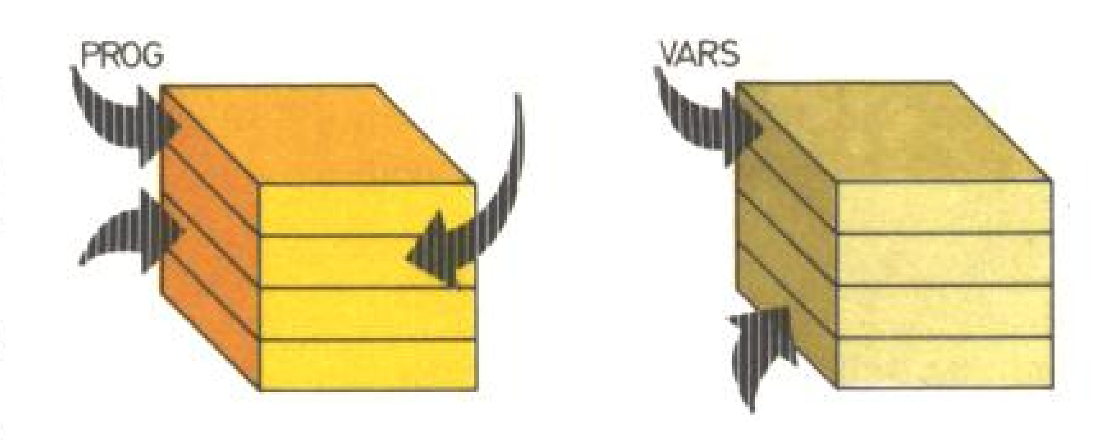
Contiene la dirección del siguiente carácter a ser interpretado por el intérprete Basic.

Puede utilizarse para enviar parámetros a una rutina código máquina llamada de la forma:

USR n: REM xxxxxxxxxxx

— X PTR IY + 37 5C5FH 23647d 2 bytes

Contiene la dirección en la cual el intérprete Basic ha encontrado un error de sintaxis.



— WORKSP IY + 39 5C61H 23649d 2 by.

Contiene la dirección del espacio temporal de trabajo utilizado por la instrucción INPUT (2089H).

— STKBOT IY + 41 5C63H 23651d 2 by.

Contiene la dirección del comienzo del stack del calculador utilizado para almacenar números en el formato de coma flotante.

— STKEND IY + 43 5C65H 23653d 2 by.

Final del calculador. Contiene la dirección de comienzo de la memoria libre.

Punteros de línea.

NEWPPC	PPC		OLDPPC
NSPPC	SUBPPC		OSPPC
EPPC		S-TOP	
ERR-NR	ERR-SP		X-PTR

— NEWPPC IY + 8 5C42H 23618d 2 by.

Contiene el número de la próxima línea que se debe ejecutar. Es utilizada por las rutinas LD-CONTRL (0808H), FOR (1D03H), y GO-TO (1E67H).

— NSPPC IY + 10 5C44H 23620d 1byte

Contiene el número de instrucción de la próxima línea que se debe ejecutar. Puede usarse en conjunción con NEWPPC para provocar un salto en el programa.

— PPC IY + 11 5C45H 23621d 2 bytes

Contiene el número de línea de la instrucción que se está ejecutando. Es usada por los comandos FOR (1D03H) y GO-SUB (1EEDH) para guardarla junto con SUBPPC bajo el stack. Siendo recuperadas por NEXT y RETURN.

— SUB-PPC IY + 13 5C47H 23623d 1 by.

Contiene el número de instrucción que se está ejecutando. Es usada en conjunción con PPC.

— E PPC IY + 15 5C49H 23625d 2 bytes

Contiene la dirección de la línea marcada con el cursor. Es usada por la rutina del comando EDIT (0FAH) y las rutinas AUTO-LIST (1795H), L LIST (17F5H) y LIST (17F9H).

— S-TOP IY + 50 5C6CH 23660d 2 bytes

Contiene la dirección del número de la primera línea que ha de ser listada por un listado automático. Es usada por la rutina AUTO-LIST (1795H).

— OLDPPC IY + 52 5C6EH 23662d 2 by.

Contiene la primera línea que debe ser interpretada mediante la instrucción CONTINUE (1E5FH).

El bucle principal MAIN-5-9 (133CH) coloca en esta variable el valor de NEWPCC o PCC según deba repetirse la última instrucción o no.

— OSPPC IY + 54 5C70H 23664d 1 byte

Contiene la primera instrucción dentro de la línea señalada por OLDPPC que debe ser interpretada mediante la instrucción CONTINUE (1E5FH).

El bucle principal MAIN-5-9 (133CH) coloca en esta variable el valor de NSPCC o SUBPCC según deba repetirse la última instrucción o no.

- **Variables de error:**

— ERR-NR IY + 0 5C3AH 23610d 1byte

Una unidad menos que el código de error generado. Si no hay error contiene 255d (FFH), que corresponde al mensaje "0 OK". Es asignada por la rutina de gestión de error ERROR-3 (0055H), y la utiliza el bucle principal MAIN-4-9 (1303H) para escribir el mensaje adecuado.

— ERR-SP IY + 3 5C3DH 23613d 2 bytes

Dirección del stack donde se encuentra la dirección de la rutina que debe ejecutarse tras la detección de un error. Normalmente es 1303H, rutina MAIN4 dentro del bucle principal. El programador puede cambiarla para hacer rutinas tipo ON ERROR ... .

— X-PTR IY + 37 5C5FH 23647d 2 bytes

Dirección donde el intérprete Basic ha detectado el error. Es leída de CH-ADD (IY + 35) por la rutina ERROR-1 (0008H).

Entre las variables del sistema hay una serie de ellas que almacenan datos referentes al teclado y los caracteres leídos:

KSTATE	REPDEL	RASP
LASTK	REPPER	PIP
MODE	K-DATA	TVDATA

— KSTATE IY-58 5C00H 23552d 8 bytes

La rutina KEYBOARD (02BFH), llamada por las interrupciones enmascarables, barre el teclado y almacena la lectura en esta variable cada vez que se realiza una interrupción.

La variable está dividida en dos zonas de 4 bytes. La zona que se va a usar depende del estado de la otra.

En el primer byte se sitúa el valor en CAPS SHIFT de la tecla actualmente pulsada. En caso, contrario FFH (255), indicando que la zona está libre de uso.

En el segundo byte se sitúa la cuenta atrás, que a su fin hará que la zona quede libre.

En el tercero, se sitúa el intervalo de repetición de las teclas.

Y en el cuarto byte, el código ASCII de la tecla pulsada.

Cuando la cuenta atrás llega a 0 los otros 4 bytes realizan esta función.

El sentido de todo esto es que se respeten los retardos de repetición de teclas REPDEL y REPPER.

— LASTK IY-50 5C08H 23560d 1 byte

Contiene el código de la última tecla pulsada. Es actualizada por KEYBOARD (02BFH).

— REPDEL IY-49 5C09H 23561d 1byte

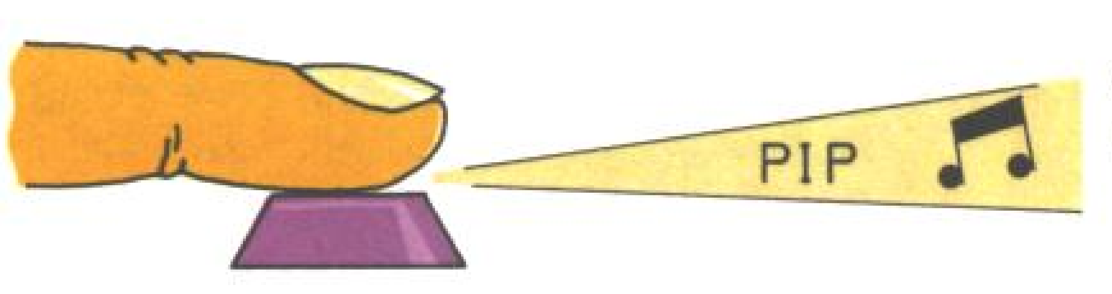
Contiene el intervalo máximo que una tecla puede mantenerse pulsada antes de que empiece a repetirse. La rutina START-NEW (11CBH) le asigna el valor 23H (0.7 segundos).

— REPPER IY-48 5C0AH 23562d 1 byte

Contiene la duración de la repetición cuando la tecla sigue siendo pulsada. La rutina START-NEW (11CBH) le asigna el valor 5 (0.1 segundos).

— RASP IY-2 5C38H 23608d 1byte

Contiene la duración del zumbido que se produce en la rutina de error del EDITOR (0F2CH).



— PIP IY-1 5C39H 23609d 1 byte

Controla la duración del sonido que produce el EDITOR (0F2CH) al admitir un carácter.

— MODE IY+7 5C41H 23617d 1 byte

Contiene el código de la letra (E,C,K,L o G) que identifica el modo en el que se está trabajando.

Es utilizada por las rutinas KEYBOARD (02BFH), EDITOR (0F2CH), ADD-CHAR (0F81H) y OUT-CURS (18E1H).

● **Variables de almacenamiento temporal:**

— K-DATA IY-45 5C0DH 23565d 1 byte

Contiene temporalmente el parámetro de un carácter de control de color. Es utilizada por la rutina KEY-INPUT (10A8H).

— TV-DATA IY-44 5C0EH 23566d 2 byte

Contiene temporalmente un carácter de control, y su primer operando, si lleva 2, hasta que sea leído el último operando en las rutinas PO-2-OPER (0A75H) y PO-1-OPER (0A7AH).

Presentamos las variables de uso general que completan la serie de variables del sistema.

DEFADD	T-ADDR	BREG
STRMS	UDG	MEM
CHARS	RAMTOP	MEMBOT
LIST-SP	P—RAMPT	

— DEFADD IY-47 5C0BH 23563d 2 bytes

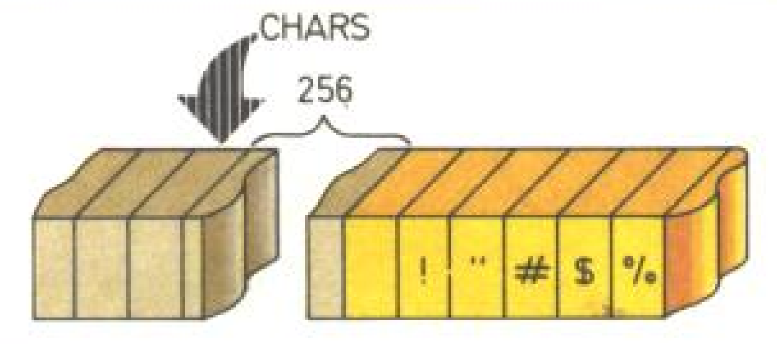
Dirección del argumento de una función definida por una instrucción DEF FN. Es usada por la instrucción FN (27BDH).

— STRMS IY-42 5C10H 23568d 38 bytes

Contiene en sus primeros 14 bytes las direcciones de los canales —3 a +3, en dos bytes cada uno. Los restantes se utilizan cuando los flujos extra están abiertos.

— CHARS IY-4 5C36H 23606d 2 bytes

Contiene la dirección del comienzo del juego de caracteres menos 256. Utilizada por RST 10H en PO-CHAR (0B65H).



— LIST SP IY+5 5C3FH 23615d 2 bytes

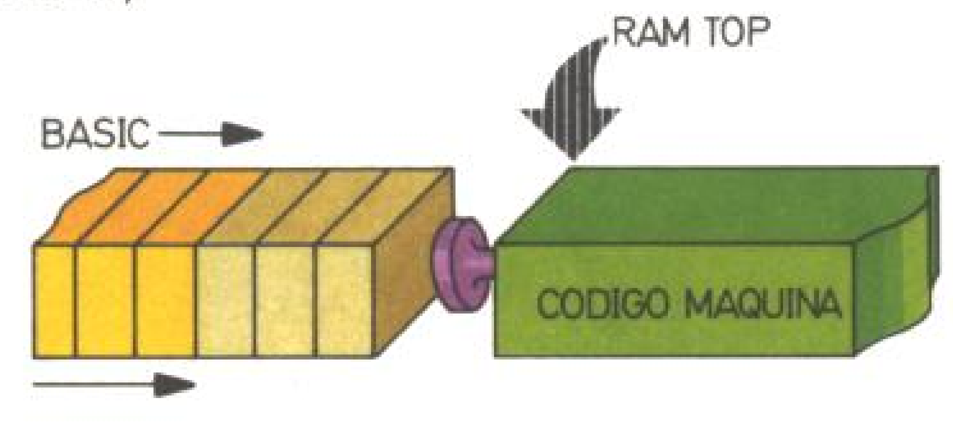
Contiene la dirección del STACK POINTER para ser llamado después de un listado. Es utilizada por las rutinas PO-SCR (0C55H) y AUTO—LIST (1795H).

— T-ADDR IY+58 5C74H 23668d 2 by.

Contiene la dirección del siguiente elemento de la tabla sintáctica situada en la dirección (1A48H).

— **UDG IY+65 5C7BH 23675d 2 bytes**

Dirección de los caracteres definidos por el usuario. Es usada por RST 10H en PO-T&UDG (0B52H).



— **RAMTOP IY+120 5CB2H 23730d 2 by.**

Dirección del último byte que puede ser usado por el Basic y el sistema. Puede modificarse con la instrucción CLEAR (1EACH) para dejar sitio a los programas en código máquina.

— **P-RAMPT IY+122 5CB4H 23732d 2 by.**

Dirección del último octeto de la memoria viva (32767 para 16Kb y 65535 para 48Kb). Es asignada por la rutina START/NEW (11CBH), señalando al último byte que funcione correctamente.

● **Variables del calculador:**

— **BREG IY+45 5C67H 23655d 1 byte**

Esta variable es utilizada por el CALCULADOR (335BH) para guardar el registro B, y ser usado por una rutina seudo-DJNZ por el generador de series en la rutina "dec-jr-nz" (367AH).

— **MEM IY+46 5C68H 23656d 2 bytes**

Señala el comienzo del área de memoria del calculador, generalmente MEMBOT. Es utilizada por la rutina del comando FOR (1D03H).

— **MEMBOT IY+88 5C92H 23698d 30 by.**

Lugar donde sitúa el CALCULADOR las 6 memorias en coma flotante mem-0 a mem-5.

Conversión HEX – DEC

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	256	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
1	4096	4352	4608	4864	5120	5376	5632	5888	6144	6400	6656	6912	7168	7424	7680	7936
2	8192	8448	8704	8960	9216	9472	9728	9984	10240	10496	10752	11008	11264	11520	11776	12032
3	12288	12544	12800	13056	13312	13568	13824	14080	14336	14592	14848	15104	15360	15616	15872	16128
4	16384	16640	16896	17152	17408	17664	17920	18176	18432	18688	18944	19200	19456	19712	19968	20224
5	20480	20736	20992	21248	21504	21760	22016	22272	22528	22784	23040	23296	23552	23808	24064	24320
6	24576	24832	25088	25344	25600	25856	26112	26368	26624	26880	27136	27392	27648	27904	28160	28416
7	28672	28928	29184	29440	29696	29952	30208	30464	30720	30976	31232	31488	31744	32000	32256	32512
8	32768	33024	33280	33536	33792	34048	34304	34560	34816	35072	35328	35584	35840	36096	36352	36608
9	36864	37120	37376	37632	37888	38144	38400	38656	38912	39168	39424	39680	39936	40192	40448	40704
A	40960	41216	41472	41728	41984	42240	42496	42752	43008	43264	43520	43776	44032	44288	44544	44800
B	45056	45312	45568	45824	46080	46336	46592	46848	47104	47360	47616	47872	48128	48384	48640	48896
C	49152	49408	49664	49920	50176	50432	50688	50944	51200	51456	51712	51968	52224	52480	52736	52992
D	53248	53504	53760	54016	54272	54528	54784	55040	55296	55552	55808	56064	56320	56576	56832	57088
E	57344	57600	57856	58112	58368	58624	58880	59136	59392	59648	59904	60160	60416	60672	60928	61184
F	61440	61696	61952	62208	62464	62720	62976	63232	63488	63744	64000	64256	64512	64768	65024	65280

El código **ASCII** (Amercian Standard Code for Information Interchange), es la representación de las funciones o caracteres más usuales en informática, acordado por la mayoría de los fabricantes, en un rango de 7 bits.

Aunque con ligeras adaptaciones para cada ordenador o cada país (el **ASCII** no incluye la ñ, por ejemplo), básicamente está aceptado que los 32 primeros códigos son de control, y el resto caracteres imprimibles.

ALTO / BAJO		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P	\	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	|	k	}
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	–	=	M	|	m	{
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	VS	/	?	O	–	o	DEL

Los 32 caracteres de control son:

Códigos típicos de Transmisión:

00	NUL	Carácter nulo (todo ceros)
01	SOH	Comienzo de cabecera
02	STX	Comienzo de texto
03	ETX	Final de texto
04	EOT	Fin de transmisión
05	ENQ	Petición de identidad
06	ACK	Reconocimiento positivo
07	BEL	Señal acústica

Códigos de control de impresión:

08	BS	Paso atrás
09	HT	Tabulación Horizontal.
0A	LF	Avance de línea
0B	VT	Tabulación vertical
OC	FF	Avance de página
OD	CR	Retorno de carro

Códigos de propósito general:

OE	SO	Salir del Estándar
OF	SI	Entrar al Estándar
10	DLE	Ampliación de control
11	DC1	Control Periférico 1
12	DC2	" " 2
13	DC3	" " 3
14	DC4	" " 4
15	NAK	Reconocimiento Negativo
16	SYN	Toma de sincronismo
17	ETB	Fin de bloque
18	CAN	Cancelación de lo anterior
19	EM	Fin de trabajo
1A	SUB	Sustituir carácter erróneo
1B	ESC	Ampliación de código
1C	FS	Separador de fichero
1D	GS	Separador de grupo
1E	RS	Separador de registro
1F	US	Separador de unidad

Códigos de designación especial:

20	SP	Espacio en blanco
7F	DEL	Borrado del último carácter

Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres
0	00	No utilizados	32	20	espacio	64	40	@	96	60	£
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	"	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	$	68	44	D	100	64	d
5	05		37	25	%	69	45	E	101	65	e
6	06	PRINT coma	38	26	&	70	46	F	102	66	f
7	07	EDIT	39	27	'	71	47	G	103	67	g
8	08	Cursor izqda.	40	28	(	72	48	H	104	68	h
9	09	Cursor dcha.	41	29	)	73	49	I	105	69	i
10	0A	Cursor abajo	42	2A	*	74	4A	J	106	6A	j
11	0B	Cursor arriba	43	2B	+	75	4B	K	107	6B	k
12	0C	DELETE	44	2C	,	76	4C	L	108	6C	l
13	0D	ENTER	45	2D	–	77	4D	M	109	6D	m
14	0E	número	46	2E	.	78	4E	N	110	6E	n
15	0F	No utilizado	47	2F	/	79	4F	O	111	6F	o
16	10	INK control	48	30	0	80	50	P	112	70	p
17	11	PAPER control	49	31	1	81	51	Q	113	71	q
18	12	FLASH control	50	32	2	82	52	R	114	72	r
19	13	BRIGHT contr.	51	33	3	83	53	S	115	73	s
20	14	INVERSE contr.	52	34	4	84	54	T	116	74	t
21	15	OVER control	53	35	5	85	55	U	117	75	u
22	16	AT control	54	36	6	86	56	V	118	76	v
23	17	TAB control	55	37	7	87	57	W	119	77	w
24	18	No utilizados	56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	\	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	↑	126	7E	~
31	1F		63	3F	?	95	5F	_	127	7F	©

Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres	Dec.	Hexa.	Caracteres
128	80	□	160	A0	(q) Gráficos def.	192	C0	USR	224	E0	LPRINT
129	81	▝	161	A1	(r)	193	C1	STR$	225	E1	LLIST
130	82	▘	162	A2	(s)	194	C2	CHR$	226	E2	STOP
131	83	▀	163	A3	(t)	195	C3	NOT	227	E3	READ
132	84	▗	164	A4	(u)	196	C4	BIN	228	E4	DATA
133	85	▐	165	A5	RND	197	C5	OR	229	E5	RESTORE
134	86	▚	166	A6	INKEY$	198	C6	AND	230	E6	NEW
135	87	▜	167	A7	PI	199	C7	<=	231	E7	BORDER
136	88	▖	168	A8	FN	200	C8	>=	232	E8	CONTINUE
137	89	▞	169	A9	POINT	201	C9	<>	233	E9	DIM
138	8A	▌	170	AA	SCREEN$	202	CA	LINE	234	EA	REM
139	8B	▛	171	AB	ATTR	203	CB	THEN	235	EB	FOR
140	8C	▄	172	AC	AT	204	CC	TO	236	EC	GO TO
141	8D	▟	173	AD	TAB	205	CD	STEP	237	ED	GO SUB
142	8E	▙	174	AE	VAL$	206	CE	DEF FN	238	EE	INPUT
143	8F	█	175	AF	CODE	207	CF	CAT	239	EF	LOAD
144	90	(a) Gráficos definibles	176	B0	VAL	208	D0	FORMAT	240	F0	LIST
145	91	(b)	177	B1	LEN	209	D1	MOVE	241	F1	LET
146	92	(c)	178	B2	SIN	210	D2	ERASE	242	F2	PAUSE
147	93	(d)	179	B3	COS	211	D3	OPEN #	243	F3	NEXT
148	94	(e)	180	B4	TAN	212	D4	CLOSE #	244	F4	POKE
149	95	(f)	181	B5	ASN	213	D5	MERGE	245	F5	PRINT
150	96	(g)	182	B6	ACS	214	D6	VERIFY	246	F6	PLOT
151	97	(h)	183	B7	ATN	215	D7	BEEP	247	F7	RUN
152	98	(i)	184	B8	LN	216	D8	CIRCLE	248	F8	SAVE
153	99	(j)	185	B9	EXP	217	D9	INK	249	F9	RANDOMIZE
154	9A	(k)	186	BA	INT	218	DA	PAPER	250	FA	IF
155	9B	(l)	187	BB	SQR	219	DB	FLASH	251	FB	CLS
156	9C	(m)	188	BC	SGN	220	DC	BRIGHT	252	FC	DRAW
157	9D	(n)	189	BD	ABS	221	DD	INVERSE	253	FD	CLEAR
158	9E	(o)	190	BE	PEEK	222	DE	OVER	254	FE	RETURN
159	9F	(p)	191	BF	IN	223	DF	OUT	255	FF	COPY

El intérprete BASIC utiliza una serie de variables para el almacenamiento temporal de datos. Estas pueden ser manejadas por un programa con las debidas precauciones según el tipo de que se trate:

- **N** El sistema cambia inmediatamente el valor.
- **A** Puede ser modificada sin problema.
- **X** Es peligroso alterarla.

INDEX	HEX	DEC	BYTES	TIPO	VARIABLE
IY−58	5C00	23552	8	N	KSTATE
IY−50	5C08	23560	1	N	LAST-K
IY−49	5C09	23561	1	A	REPDEL
IY−48	5C0A	23562	1	A	REPPER
IY−47	5C0B	23563	2	N	DEFADD
IY−45	5C0D	23565	1	N	K-DATA
IY−44	5C0E	23566	2	N	TVDATA
IY−42	5C10	23568	38	X	STRMS
IY−4	5C36	23606	2	A	CHARS
IY−2	5C38	23608	1	A	RASP

INDEX	HEX	DEC	BYTES	TIPO	VARIABLE
IY−1	5C39	23609	1	A	PIP
IY+0	5C3A	23610	1	A	ERR-NR
IY+1	5C3B	23611	1	X	FLAGS
IY+2	5C3C	23612	1	X	TV-FLAG
IY+3	5C3D	23613	2	X	ERR-SP
IY+5	5C3F	23615	2	N	LIST-SP
IY+7	5C41	23617	1	N	MODE
IY+8	5C42	23618	2	A	NEWPPC
IY+10	5C44	23620	1	A	NSPPC
IY+11	5C45	23621	2	A	PPC
IY+13	5C47	23623	1	A	SUBPPC
IY+14	5C48	23624	1	A	BORDCR
IY+15	5C49	23625	2	A	E-PPC
IY+17	5C4B	23627	2	X	VARS
IY+19	5C4D	23629	2	N	DEST
IY+21	5C4F	23631	2	X	CHANS
IY+23	5C51	23633	2	X	CURCHL
IY+25	5C53	23635	2	X	PROG
IY+27	5C55	23637	2	X	NXTLIN
IY+29	5C57	23639	2	X	DATADD

INDEX	HEX	DEC	BYTES	TIPO	VARIABLE
IY+31	5C59	23641	2	X	E-LINE
IY+33	5C5B	23643	2	A	K-CUR
IY+35	5C5D	23645	2	X	CH-ADD
IY+37	5C5F	23647	2	A	X-PTR
IY+39	5C61	23649	2	X	WORKSP
IY+41	5C63	23651	2	X	STKBOT
IY+43	5C65	23653	2	X	STKEND
IY+45	5C67	23655	1	N	BREG
IY+46	5C68	23656	2	N	MEM
IY+48	5C6A	23658	1	A	FLAGS2
IY+49	5C6B	23659	1	X	DF-SZ
IY+50	5C6C	23660	2	A	S-TOP
IY+52	5C6E	23662	2	A	OLDPPC
IY+54	5C70	23664	1	A	OSPPC
IY+55	5C71	23665	1	N	FLAGX
IY+56	5C72	23666	2	N	STRLEN
IY+58	5C74	23668	2	N	T-ADDR
IY+60	5C76	23670	2	A	SEED
IY+62	5C78	23672	3	A	FRAMES
IY+65	5C7B	23675	2	A	UDG

INDEX	HEX	DEC	BYTES	TIPO	VARIABLE
IY+67	5C7D	23677	2	A	COORDS
IY+69	5C7F	23679	1	A	P-POSN
IY+70	5C80	23680	1	A	PR-CC
IY+71	5C81	23681	1	A	No usada
IY+72	5C82	23682	2	A	ECHO-E
IY+74	5C84	23684	2	A	DF-CC
IY+76	5C86	23686	2	A	DFCCL
IY+78	5C88	23688	2	X	S-POSN
IY+80	5C8A	23690	2	X	SPOSNL
IY+82	5C8C	23692	1	A	SCR-CT
IY+83	5C8D	23693	1	A	ATTR-P
IY+84	5C8E	23694	1	A	MASK-P
IY+85	5C8F	23695	1	N	ATTR-T
IY+86	5C90	23696	1	N	MASK-T
IY+87	5C91	23697	1	A	P-FLAG
IY+88	5C92	23698	30	N	MEMBOT
IY+118	5CB0	23728	2	A	No usada
IY+120	5CB2	23730	2	A	RAMTOP
IY+122	5CB4	23732	2	A	P-RAMT

Instrucciones sin prefijo:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 NOP	1 LD BC, NN	2 LD (BC), A	3 INC BC	4 INC B	5 DEC B	6 LD B, N	7 RLCA	8 EX AF, AF'	9 ADD HL, BC	10 LD A, (BC)	11 DEC BC	12 INC C	13 DEC C	14 LD C, N	15 RRCA
1	16 DJNZ DIS	17 LD DE, NN	18 LD (DE), A	19 INC DE	20 INC D	21 DEC D	22 LD D, N	23 RLA	24 JR DIS	25 ADD HL, DE	26 LD A, (DE)	27 DEC DE	28 INC E	29 DEC E	30 LD E, N	31 RRA
2	32 JR NZ, DIS	33 LD HL, NN	34 LD (NN), HL	35 INC HL	36 INC H	37 DEC H	38 LD H, N	39 DAA	40 JR Z, DIS	41 ADD HL, HL	42 LD HL, (NN)	43 DEC HL	44 INC L	45 DEC L	46 LD L, N	47 CPL
3	48 JR NC, DIS	49 LD SP, NN	50 LD (NN), A	51 INC SP	52 INC (HL)	53 DEC (HL)	54 LD (HL), N	55 SCF	56 JR C, DIS	57 ADD HL, SP	58 LD A, (NN)	59 DEC SP	60 INC A	61 DEC A	62 LD A, N	63 CCF
4	64 LD B, B	65 LD B, C	66 LD B, D	67 LD B, E	68 LD B, H	69 LD B, L	70 LD B, (HL)	71 LD B, A	72 LD C, B	73 LD C, C	74 LD C, D	75 LD C, E	76 LD C, H	77 LD C, L	78 LD C, (HL)	79 LD C, A
5	80 LD D, B	81 LD D, C	82 LD D, D	83 LD D, E	84 LD D, H	85 LD D, L	86 LD D, (HL)	87 LD D, A	88 LD E, B	89 LD E, C	90 LD E, D	91 LD E, E	92 LD E, H	93 LD E, L	94 LD E, (HL)	95 LD E, A
6	96 LD H, B	97 LD H, C	98 LD H, D	99 LD H, E	100 LD H, H	101 LD H, L	102 LD H, (HL)	103 LD H, A	104 LD L, B	105 LD L, C	106 LD L, D	107 LD L, E	108 LD L, H	109 LD L, L	110 LD L, (HL)	111 LD L, A
7	112 LD (HL), B	113 LD (HL), C	114 LD (HL), D	115 LD (HL), E	116 LD (HL), H	117 LD (HL), L	118 HALT	119 LD (HL), A	120 LD A, B	121 LD A, C	122 LD A, D	123 LD A, E	124 LD A, H	125 LD A, L	126 LD A, (HL)	127 LD A, A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	128 ADD A, B	129 ADD A, C	130 ADD A, D	131 ADD A, E	132 ADD A, H	133 ADD A, L	134 ADD A, (HL)	135 ADD A, A	136 ADC A, B	137 ADC A, C	138 ADC A, D	139 ADC A, E	140 ADC A, H	141 ADC A, L	142 ADC A, (HL)	143 ADC A, A
9	144 SUB B	145 SUB C	146 SUB D	147 SUB E	148 SUB H	149 SUB L	150 SUB (HL)	151 SUB A	152 SBC, A, B	153 SBC A, C	154 SBC A, D	155 SBC A, E	156 SBC A, H	157 SBC A, L	158 SBC A, (HL)	159 SBC A, A
A	160 AND B	161 AND C	162 AND D	163 AND E	164 AND H	165 AND L	166 AND (HL)	167 AND A	168 XOR B	169 XOR C	170 XOR D	171 XOR E	172 XOR H	173 XOR L	174 XOR (HL)	175 XOR A
B	176 OR B	177 OR C	178 OR D	179 OR E	180 OR H	181 OR L	182 OR (HL)	183 OR A	184 CP B	185 CP C	186 CP D	187 CP E	188 CP H	189 CP L	190 CP (HL)	191 CP A
C	192 RET NZ	193 POP BC	194 JP NZ, NN	195 JP NN	196 CALL NZ, NN	197 PUSH BC	198 ADD A, N	199 RST O	200 RET Z	201 RET	202 JP Z, NN	203 prefijo	204 CALL Z, NN	205 CALL NN	206 ADC A, N	207 RST 8
D	208 RET NC	209 POP DE	210 JP NC, NN	211 OUT (N), A	212 CALL NC, NN	213 PUSH DE	214 SUB N	215 RST IOH	216 RET C	217 EXX	218 JP C, NN	219 IN A, (N)	220 CALL C, NN	221 prefijo	222 SBC A, N	223 RST 18H
E	224 RET PO	225 POP HL	226 JP PO, NN	227 EX (SP), HL	228 CALL PO, NN	229 PUSH HL	230 AND N	231 RST 20H	232 RET PE	233 JP (HL)	234 JP PE, NN	235 EX DE, HL	236 CALL PE, NN	237 prefijo	238 XOR N	239 RST 28H
F	240 RET P	241 POP AF	242 JP P, NN	243 DI	244 CALL P, NN	245 PUSH AF	246 OR N	247 RST 30H	248 RET M	249 LD SP, HL	250 JP M, NN	251 EI	252 CALL M, NN	253 prefijo	254 CP N	255 RST 38H

Instrucciones con prefijo CB:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 RLC B	1 RLC C	2 RLC D	3 RLC E	4 RLC H	5 RLC L	6 RLC(HL)	7 RLC A	8 RRC B	9 RRC C	10 RRC D	11 RRC E	12 RRC H	13 RRC L	14 RRC (HL)	15 RRC A
1	16 RL B	17 RL C	18 RL D	19 RL E	20 RL H	21 RL L	22 RL (HL)	23 RL A	24 RR B	25 RR C	26 RR D	27 RR E	28 RR H	29 RR L	30 RR (HL)	31 RR A
2	32 SLA B	33 SLA C	34 SLA D	35 SLA E	36 SLA H	37 SLA L	38 SLA (HL)	39 SLA A	40 SRA B	41 SRA C	42 SRA D	43 SRA E	44 SRA H	45 SRA L	46 SRA (HL)	47 SRA A
3	48	49	50	51	52	53	54	55	56 SRL B	57 SRL C	58 SRL D	59 SRL E	60 SRL H	61 SRL L	62 SRL (HL)	63 SRL A
4	64 BIT 0, B	65 BIT 0, C	66 BIT 0, D	67 BIT 0, E	68 BIT 0, H	69 BIT 0, L	70 BIT 0, (HL)	71 BIT 0, A	72 BIT 1, B	73 BIT 1, C	74 BIT 1, D	75 BIT 1, E	76 BIT 1, H	77 BIT 1, L	78 BIT 1, (HL)	79 BIT 1, A
5	80 BIT 2, B	81 BIT 2, C	82 BIT 2, D	83 BIT 2, E	84 BIT 2, H	85 BIT 2, L	86 BIT 2, (HL)	87 BIT 2, A	88 BIT 3, B	89 BIT 3, C	90 BIT 3, D	91 BIT 3, E	92 BIT 3, H	93 BIT 3, L	94 BIT 3, (HL)	95 BIT 3, A
6	96 BIT 4, B	97 BIT 4, C	98 BIT 4, D	99 BIT 4, E	100 BIT 4, H	101 BIT 4, L	102 BIT 4, (HL)	103 BIT 4, A	104 BIT 5, B	105 BIT 5, C	106 BIT 5, D	107 BIT 5, E	108 BIT 5, H	109 BIT 5, L	110 BIT 5, (HL)	111 BIT 5, A
7	112 BIT 6, B	113 BIT 6, C	114 BIT 6, D	115 BIT 6, E	116 BIT 6, H	117 BIT 6, L	118 BIT 6, (HL)	119 BIT 6, A	120 BIT 7, B	121 BIT 7, C	122 BIT 7, D	123 BIT 7, E	124 BIT 7, H	125 BIT 7, L	126 BIT 7, (HL)	127 BIT 7, A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	128 RES 0, B	129 RES 0, C	130 RES 0, D	131 RES 0, E	132 RES 0, H	133 RES 0, L	134 RES 0, (HL)	135 RES 0, A	136 RES 1, B	137 RES 1, C	138 RES 1, D	139 RES 1, E	140 RES 1, H	141 RES 1, L	142 RES 1, (HL)	143 RES 1, A
9	144 RES 2, B	145 RES 2, C	146 RES 2, D	147 RES 2, E	148 RES 2, H	149 RES 2, L	150 RES 2, (HL)	151 RES 2, A	152 RES 3, B	153 RES 3, C	154 RES 3, D	155 RES 3, E	156 RES 3, H	157 RES 3, L	158 RES 3, (HL)	159 RES 3, A
A	160 RES 4, B	161 RES 4, C	162 RES 4, D	163 RES 4, E	164 RES 4, H	165 RES 4, L	166 RES 4, (HL)	167 RES 4, A	168 RES 5, B	169 RES 5, C	170 RES 5, D	171 RES 5, E	172 RES 5, H	173 RES 5, L	174 RES 5, (HL)	175 RES 5, A
B	176 RES 6, B	177 RES 6, C	178 RES 6, D	179 RES 6, E	180 RES 6, H	181 RES 6, L	182 RES 6, (HL)	183 RES 6, A	184 RES 7, B	185 RES 7, C	186 RES 7, D	187 RES 7, E	188 RES 7, H	189 RES 7, L	190 RES 7, (HL)	191 RES 7, A
C	192 SET 0, B	193 SET 0, C	194 SET 0, D	195 SET 0, E	196 SET 0, H	197 SET 0, L	198 SET 0, (HL)	199 SET 0, A	200 SET 1, B	201 SET 1, C	202 SET 1, D	203 SET 1, E	204 SET 1, H	205 SET 1, L	206 SET 1, (HL)	207 SET 1, A
D	208 SET 2, B	209 SET 2, C	210 SET 2, D	211 SET 2, E	212 SET 2, H	213 SET 2, L	214 SET 2, (HL)	215 SET 2, A	216 SET 3, B	217 SET 3, C	218 SET 3, D	219 SET 3, E	220 SET 3, H	221 SET 3, L	222 SET 3, (HL)	223 SET 3, A
E	224 SET 4, B	225 SET 4, C	226 SET 4, D	227 SET 4, E	228 SET 4, H	229 SET 4, L	230 SET 4, (HL)	231 SET 4, A	232 SET 5, B	233 SET 5, C	234 SET 5, D	235 SET 5, E	236 SET 5, H	237 SET 5, L	238 SET 5, (HL)	239 SET 5, A
F	240 SET 6, B	241 SET 6, C	242 SET 6, D	243 SET 6, E	244 SET 6, H	245 SET 6, L	246 SET 6, (HL)	247 SET 6, A	248 SET 7, B	249 SET 7, C	250 SET 7, D	251 SET 7, E	252 SET 7, H	253 SET 7, L	254 SET 7, (HL)	255 SET 7, A

Instrucciones con prefijo ED:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	64 IN B, (C)	65 OUT (C), B	66 SBC HL, BC	67 LD (NN), BC	68 NEG	69 RETN	70 IM 0	71 LD I, A	72 IN C, (C)	73 OUT (C), C	74 ADC HL, BC	75 LD BC, (NN)	76	77 RETI	78	79 LD R, A
5	80 IN D, (C)	81 OUT (C), D	82 SBC HL, DE	83 LD (NN), DE	84	85	86 IM 1	87 LD A, I	88 IN E, (C)	89 OUT (C), E	90 ADC HL, DE	91 LD DE, (NN)	92	93	94 IM 2	95 LD A, R
6	96 IN H, (C)	97 OUT (C), H	98 SBC HL, HL	99 LD (NN) HL	100	101	102	103 RRD	104 IN L, (C)	105 OUT (C), L	106 ADC HL, HL	107 LD HL, (NN)	108	109	110	111 RLD
7	112	113	114 SBC HL, SP	115 LD (NN), SP	116	117	118	119	120 IN A, (C)	121 OUT (C), A	122 ADC HL, SP	123 LD SP, (NN)	124	125	126	127

A	160 LDI	161 CPI	162 INI	163 OUTI	164	165	166	167	168 LDD	169 CPD	170 IND	171 OUTD	172	173	174	175
B	176 LDIR	177 CPIR	178 INIR	179 OTIR	180	181	182	183	184 LDDR	185 CPDR	186 INDR	187 OTDR	188	189	190	191

Instrucciones con prefijo DD y FD

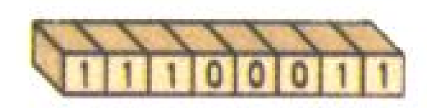
- Las instrucciones con prefijo DD se refieren al registro índice IX.
- Las instrucciones con prefijo FD se refieren al registro índice IY.

Para desensamblar dichas instrucciones deben usarse las tablas de instrucciones ordinarias, haciendo la siguiente sustitución:

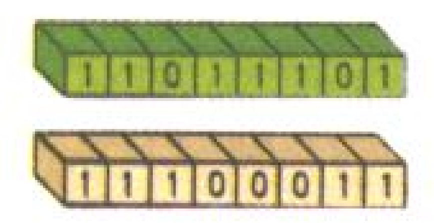
— HL debe sustituirse por IX o IY
— (HL) se sustituirá por (IX + d) o (IY + d)
Debe tenerse en cuenta que en las instrucciones de manipulación de bits el byte de desplazamiento se sitúa en penúltimo lugar.

Ejemplos:

E3H corresponde a EX (SP),HL



DDH,E3H corresponderá a EX (SP),IX

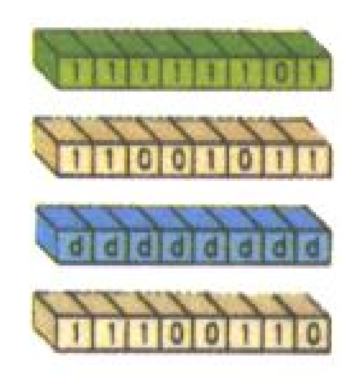


CBH,6EH corresponde a BIT 5,(HL)

1 1 0 0 1 0 1 1

1 1 1 0 0 1 1 0

FDH,CBH, d ,6EH corresponderá a BIT 5,(IY + d)



Cod.	Direc.	Comando	Clases y separadores	Rutinas
206	1AF9H	DEF-FN		5 1F60H
207	1B14H	CAT		0 1793H
208	1B06H	FORMAT	A	0 1793H
209	1B0AH	MOVE	A ,A	0 1793H
210	1B10H	ERASE	A	0 1793H
211	1AFCH	OPEN#	6 , A	0 1736H
212	1B02H	CLOSE#	6	0 16E5H
213	1AE2H	MERGE		B
214	1AE1H	VERIFY		B
215	1AE3H	BEEP	8	0 03F8H
216	1AE7H	CIRCLE	9	5 2320H
217	1AEBH	INK		7
218	1AECH	PAPER		7
219	1AEDH	FLASH		7
220	1AEEH	BRIGHT		7
221	1AEFH	INVERSE		7
222	1AF0H	OVER		7
223	1AF1H	OUT	8	0 1E7AH
224	1AD9H	LPRINT		5 1FC9H
225	1ADCH	LLIST		5 17F5H

Cod.	Direc.	Comando	Clases y separadores	Rutinas
226	1A8AH	STOP		0 1CEEH
227	1AC9H	READ		5 1DEDH
228	1ACCH	DATA		5 1E27H
229	1ACFH	RESTORE		3 1E42H
230	1AA8H	NEW		0 11B7H
231	1AF5H	BORDER	6	0 2294H
232	1AB8H	CONTINUE		0 1EF5H
233	1AA2H	DIM		5 2C02H
234	1AA5H	REM		5 1BB2H
235	1A90H	FOR	4 = 6 TO 6	5 1D03H
236	1A7DH	GO-TO	6	0 1E67H
237	1A86H	GO-SUB	6	0 1EEDH
238	1A9FH	INPUT		5 2089H
239	1AE0H	LOAD		B
240	1AAEH	LIST		5 17F9H
241	1A7AH	LET	1 =	2
242	1AC5H	PAUSE	6	0 1F3AH
243	1A98H	NEXT	4	0 1DABH
244	1AB1H	POKE	8	0 1E80H
245	1A9CH	PRINT		5 1FCDH

Cod.	Direc.	Comando	Clases y separadores		Rutinas
246	1AC1H	PLOT	9	0	22DCH
247	1AABH	RUN		3	1EA1H
248	1ADFH	SAVE		B	
249	1AB5H	RANDOMIZE		3	1E4FH
250	1A81H	IF	6 THEN	5	1CF0H
251	1ABEH	CLS		0	0D6BH
252	1AD2H	DRAW	9	5	2382H
253	1ABBH	CLEAR		3	1EACH
254	1A8DH	RETURN		0	1F23H
255	1AD6H	COPY		0	0EACH

Clase 0: Salta a la rutina sin operandos (1C10H).

Clase 1: (LET) Localiza una variable y actualiza DEST STRLEN y FLAGX (1C1FH).

Clase 2: Asigna un valor a la variable: LET 2AFFH (1C4EH).

Clase 3: Busca una expresión numérica (en su defecto entiende 0) y salta a la rutina (1C0DH).

Clase 4: Variable de un solo carácter; control FOR NEXT (1C6CH).

Clase 5: Salta a la rutina con operandos (1C11H).

Clase 6: Buca una expresión numérica (1C82H).

Clase 7: Rutinas de color: PERMS (1C96H).

Clase 8: Busca dos expresiones numéricas separadas por una coma (1C7AH).

Clase 9: Como la 8 pero pueden estar precedidas de comandos de color temporal (1CBEH).

Clase A: Busca una expresión de cadena (1C8CH).

Clase B: Rutinas de cassette (1CDB); salta a SAVE-ETC (0605H).

MNEMONICO	HEXADECIMAL	FICHA
ADC A,(HL)	8E	I-17
ADC A,(IX+d)	DD 8E XX	I-17
ADC A,(IY+d)	FD 8E XX	I-17
ADC A,A	8F	I-16
ADC A,B	88	I-16
ADC A,C	89	I-16
ADC A,D	8A	I-16
ADC A,E	8B	I-16
ADC A,H	8C	I-16
ADC A,L	8D	I-16
ADC A,n	CE XX	I-16
ADC HL,BC	ED 4A	I-29
ADC HL,DE	ED 5A	I-29
ADC HL,HL	ED 6A	I-29
ADC HL,SP	ED 7A	I-29
ADD A,(HL)	86	I-15
ADD A,(IX+d)	DD 86 XX	I-15
ADD A,(IY+d)	FD 86 XX	I-15
ADD A,A	87	I-14
ADD A,B	80	I-14
ADD A,C	81	I-14
ADD A,D	82	I-14
ADD A,E	83	I-14
ADD A,H	84	I-14
ADD A,L	85	I-14
ADD A,n	C6 XX	I-14
ADD HL,BC	09	I-28
ADD HL,DE	19	I-28
ADD HL,HL	29	I-28

MNEMONICO	HEXADECIMAL	FICHA
ADD HL,SP	39	I-28
ADD IX,BC	DD 09	I-28
ADD IX,DE	DD 19	I-28
ADD IX,IX	DD 29	I-28
ADD IX,SP	DD 39	I-28
ADD IY,BC	FD 09	I-28
ADD IY,DE	FD 19	I-28
ADD IY,IY	FD 29	I-28
ADD IY,SP	FD 39	I-28
AND (HL)	A6	I-22
AND (IX+d)	DD A6 XX	I-22
AND (IY+d)	FD A6 XX	I-22
AND A	A7	I-22
AND B	A0	I-22
AND C	A1	I-22
AND D	A2	I-22
AND E	A3	I-22
AND H	A4	I-22
AND L	A5	I-22
AND n	E6 XX	I-22
BIT 0,(HL)	CB 46	I-50
BIT 0,(IX+d)	DD CB XX 46	I-50
BIT 0,(IY+d)	FD CB XX 46	I-50
BIT 0,A	CB 47	I-49
BIT 0,B	CB 40	I-49
BIT 0,C	CB 41	I-49
BIT 0,D	CB 42	I-49
BIT 0,E	CB 43	I-49
BIT 0,H	CB 44	I-49

MNEMONICO	HEXADECIMAL	FICHA
BIT 0,L	CB 45	I-49
BIT 1,(HL)	CB 4E	I-50
BIT 1,(IX+d)	DD CB XX 4E	I-50
BIT 1,(IY+d)	FD CB XX 4E	I-50
BIT 1,A	CB 4F	I-49
BIT 1,B	CB 48	I-49
BIT 1,C	CB 49	I-49
BIT 1,D	CB 4A	I-49
BIT 1,E	CB 4B	I-49
BIT 1,H	CB 4C	I-49
BIT 1,L	CB 4D	I-49
BIT 2,(HL)	CB 56	I-50
BIT 2,(IX+d)	DD CB XX 56	I-50
BIT 2,(IY+d)	FD CB XX 56	I-50
BIT 2,A	CB 57	I-49
BIT 2,B	CB 50	I-49
BIT 2,C	CB 51	I-49
BIT 2,D	CB 52	I-49
BIT 2,E	CB 53	I-49
BIT 2,H	CB 54	I-49
BIT 2,L	CB 55	I-49
BIT 3,(HL)	CB 5E	I-50
BIT 3,((IX+d)	DD CB XX 5E	I-50
BIT 3,(IY+d)	FD CB XX 5E	I-50
BIT 3,A	CB 5F	I-49
BIT 3,B	CB 58	I-49
BIT 3,C	CB 59	I-49
BIT 3,D	CB 5A	I-49
BIT 3,E	CB 5B	I-49

MNEMONICO	HEXADECIMAL	FICHA	MNEMONICO	HEXADECIMAL	FICHA	MNEMONICO	HEXADECIMAL	FICHA
BIT 3,H	CB 5C	I-49	BIT 6,E	CB 73	I-49	CP D	BA	I-25
BIT 3,L	CB 5D	I-49	BIT 6,H	CB 74	I-49	CP E	BB	I-25
BIT 4,(HL)	CB 66	I-50	BIT 6,L	CB 75	I-49	CP H	BC	I-25
BIT 4,(IX + d)	DD CB XX 66	I-50	BIT 7,(HL)	CB 7E	I-50	CP L	BD	I-25
BIT 4,(IY + d)	FD CB XX 66	I-50	BIT 7,(IX + d)	DD CB XX 7E	I-50	CP n	FE XX	I-25
BIT 4,A	CB 67	I-49	BIT 7,(IY + d)	FD CB XX 7E	I-50	CPD	ED A9	I-37
BIT 4,B	CB 60	I-49	BIT 7,A	CB 7F	I-49	CPDR	ED B9	I-37
BIT 4,C	CB 61	I-49	BIT 7,B	CB 78	I-49	CPI	ED A1	I-36
BIT 4,D	CB 62	I-49	BIT 7,C	CB 79	I-49	CPIR	ED B1	I-36
BIT 4,E	CB 63	I-49	BIT 7,D	CB 7A	I-49	CPL	2F	I-38
BIT 4,H	CB 64	I-49	BIT 7,E	CB 7B	I-49	DAA	27	I-38
BIT 4,L	CB 65	I-49	BIT 7,H	CB 7C	I-49	DEC (HL)	35	I-27
BIT 5,(HL)	CB 6E	I-50	BIT 7,L	CB 7D	I-49	DEC (IX + d)	DD 35 XX	I-27
BIT 5,(IX + d)	DD CB XX 6E	I-50	CALL C,nn	DC XX XX	I-59	DEC (IY + d)	FD 35 XX	I-27
BIT 5,(IY + d)	FD CB XX 6E	I-50	CALL M,nn	FC XX XX	I-59	DEC A	3D	I-27
BIT 5,A	CB 6F	I-49	CALL NC,nn	D4 XX XX	I-59	DEC B	05	I-27
BIT 5,B	CB 68	I-49	CALL NZ,nn	C4 XX XX	I-59	DEC BC	0B	I-31
BIT 5,C	CB 69	I-49	CALL P,nn	F4 XX XX	I-59	DEC C	0D	I-27
BIT 5,D	CB 6A	I-49	CALL PE,nn	EC XX XX	I-59	DEC D	15	I-27
BIT 5,E	CB 6B	I-49	CALL PO,nn	E4 XX XX	I-59	DEC DE	1B	I-31
BIT 5,H	CB 6C	I-49	CALL Z,nn	CC XX XX	I-59	DEC E	1D	I-27
BIT 5,L	CB 6D	I-49	CALL nn	CD XX XX	I-59	DEC H	25	I-27
BIT 6,(HL)	CB 76	I-50	CCF	3F	I-39	DEC HL	2B	I-31
BIT 6,(IX + d)	DD CB XX 76	I-50	CP (HL)	BE	I-25	DEC IX	DD 2B	I-31
BIT 6,(IY + d)	FD CB XX 76	I-50	CP (IX + d)	DD BE XX	I-25	DEC IY	FD 2B	I-31
BIT 6,A	CB 77	I-49	CP (IY + d)	FD BE XX	I-25	DEC L	2D	I-27
BIT 6,B	CB 70	I-49	CP A	BF	I-25	DEC SP	3B	I-31
BIT 6,C	CB 71	I-49	CP B	B8	I-25	DI	F3	I-40
BIT 6,D	CB 72	I-49	CP C	B9	I-25	DJNZ,e	10 XX	I-57

MNEMONICO	HEXADECIMAL	FICHA
EI	FB	I-40
EX (SP) ,HL	E3	I-13
EX (SP) ,IX	DD E3	I-13
EX (SP) ,IY	FD E3	I-13
EX AF,AF'	08	I-12
EX DE,HL	EB	I-12
EXX	D9	I-12
HALT	76	I-39
IM 0	ED 46	I-40
IM 1	ED 56	I-40
IM 2	ED 5E	I-40
IN A,(C)	ED 78	I-62
IN A,(n)	DB XX	I-62
IN B,(C)	ED 40	I-62
IN C,(C)	ED 48	I-62
IN D,(C)	ED 50	I-62
IN E,(C)	ED 58	I-62
IN H,(C)	ED 60	I-62
IN L,(C)	ED 68	I-62
INC (HL)	34	I-26
INC (IX + d)	DD 34 XX	I-26
INC (IY + d)	FD 34 XX	I-26
INC A	3C	I-26
INC B	04	I-26
INC BC	03	I-30
INC C	0C	I-26
INC D	14	I-26
INC DE	13	I-30
INC E	1C	I-26

MNEMONICO	HEXADECIMAL	FICHA
INC H	24	I-26
INC HL	23	I-30
INC IX	DD 23	I-30
INC IY	FD 23	I-30
INC L	2C	I-26
INC SP	33	I-30
IND	ED AA	I-64
INDR	ED BA	I-64
INI	ED A2	I-63
INIR	ED B2	I-63
JP (HL)	E9	I-56
JP (IX)	DD E9	I-56
JP (IY)	FD E9	I-56
JP C,nn	DA XX XX	I-55
JP M,nn	FA XX XX	I-55
JP NC,nn	D2 XX XX	I-55
JP NZ,nn	C2 XX XX	I-55
JP P,nn	F2 XX XX	I-55
JP PE,nn	EA XX XX	I-55
JP PO,nn	E2 XX XX	I-55
JP Z,nn	CA XX XX	I-55
JP nn	C3 XX XX	I-55
JR C,e	38 XX XX	I-58
JR NC,e	30 XX	I-58
JR NZ,e	20 XX	I-58
JR Z,e	28 XX	I-58
JR e	18 xx	I-57
LD (nn) ,A	32 XX XX	I-3
LD (nn) ,BC	ED 43 XX XX	I-10

MNEMONICO	HEXADECIMAL	FICHA
LD (nn) ,DE	ED 53 XX XX	I-10
LD (nn), HL	ED 63 XX XX	I-10
LD (nn) ,HL	22 XX XX	I-10
LD (nn) ,IX	DD 22 XX XX	I-10
LD (nn) ,IY	FD 22 XX XX	I-10
LD (nn) ,SP	ED 73 XX XX	I-10
LD (BC) ,A	02	I-5
LD (DE) ,A	12	I-5
LD (HL) ,A	77	I-4
LD (HL) ,B	70	I-4
LD (HL) ,C	71	I-4
LD (HL) ,D	72	I-4
LD (HL) ,E	73	I-4
LD (HL) ,H	74	I-4
LD (HL) ,L	75	I-4
LD (HL) ,n	36 XX	I-4
LD (IX + d) ,A	DD 77 XX	I-4
LD (IX + d) ,B	DD 70 XX	I-4
LD (IX + d) ,C	DD 71 XX	I-4
LD (IX + d) ,D	DD 72 XX	I-4
LD (IX + d) ,n	DD 36 XX XX	I-4
LD (IX + d) ,E	DD 73 XX	I-4
LD (IX + d) ,H	DD 74 XX	I-4
LD (IX + d) ,L	DD 75 XX	I-4
LD (IY + d) ,A	FD 77 XX	I-4
LD (IY + d) ,B	FD 70 XX	I-4
LD (IY + d) ,C	FD 71 XX	I-4
LD (IY + d) ,D	FD 72 XX	I-4
LD (IY + d) ,E	FD 73 XX	I-4

MNEMONICO	HEXADECIMAL	FICHA
LD (IY + d) ,H	FD 74 XX	I-4
LD (IY + d) ,L	FD 75 XX	I-4
LD (IY + d) ,n	FD 36 XX XX	I-4
LD A, (BC)	0A	I-5
LD A, (DE)	1A	I-5
LD A, (HL)	7E	I-4
LD A, (IX + d)	DD 7E XX	I-4
LD A, (IY + d)	FD 7E XX	I-4
LD A, (nn)	3A XX XX	I-3
LD A,A	7F	I-1
LD A,B	78	I-1
LD A,C	79	I-1
LD A,D	7A	I-1
LD A,E	7B	I-1
LD A,H	7C	I-1
LD A,I	ED 57	I-2
LD A,L	7D	I-1
LD A,n	3E XX	I-1
LD A,R	ED 5F	I-2
LD B, (HL)	46	I-4
LD B, (IX + d)	DD 46 XX	I-4
LD B, (IY + d)	FD 46 XX	I-4
LD B,A	47	I-4
LD B,B	40	I-4
LD B,C	41	I-4
LD B,D	42	I-4
LD B,E	43	I-4
LD B,H	44	I-4
LD B,L	45	I-1

MNEMONICO	HEXADECIMAL	FICHA
LD B,n	06 XX	1-4
LD BC, (nn)	ED 4B XX XX	I-9
LD BC,nn	01 XX XX	I-8
LD C, (HL)	4E	I-4
LD C, (IX + d)	DD 4E XX	I-4
LD C, (IY + d)	FD 4E XX	I-4
LD C,A	4F	I-1
LD C,B	48	I-1
LD C,C	49	I-1
LD C,D	4A	I-1
LD C,E	4B	I-1
LD C,H	4C	I-1
LD C,L	4D	I-1
LD C,n	0E XX	I-1
LD D,(HL)	56	I-4
LD D, (IX + d)	DD 56 XX	I-4
LD D, (IY + d)	FD 56 XX	I-4
LD D,A	57	I-4
LD D,B	50	I-4
LD D,C	51	I-4
LD D,D	52	I-4
LD D,E	53	I-4
LD D,H	54	I-4
LD D,L	55	I-4
LD D,n	16 XX	I-4
LD DE, (nn)	ED 5B XX XX	I-9
LD DE,nn	11 XX XX	I-8
LD E, (HL)	5E	I-4
LD E, (IX + d)	DD 5E XX	I-4

MNEMONICO	HEXADECIMAL	FICHA
LD E, (IY + d)	FD 5E XX	I-4
LD E,A	5F	I-1
LD E,B	58	I-1
LD E,C	59	I-1
LD E,D	5A	I-1
LD E,E	5B	I-1
LD E,H	5C	I-1
LD E,L	5D	I-1
LD E,n	1E XX	I-1
LD H, (HL)	66	I-4
LD H, (IX + d)	DD 66 XX	I-4
LD H, (Y + d)	FD 66 XX	I-4
LD H,A	67	I-4
LD H,B	60	I-1
LD H,C	61	I-1
LD H,D	62	I-1
LD H,E	63	I-1
LD H,H	64	I-1
LD H,L	65	I-1
LD H,n	26 XX	I-1
LD HL, (nn)	ED 6B XX XX	I-9
LD HL, (nn)	2A XX XX	I-9
LD HL,nn	21 XX XX	I-8
LD I,A	ED 47	I-2
LD IX, (nn)	DD 2A XX XX	I-9
LD IX,nn	DD 21 XX XX	I-8
LD IY, (nn)	FD 2A XX XX	I-9
LD IY,nn	FD 21 XX XX	I-8
LD L, (HL)	6E	1-4

MNEMONICO	HEXADECIMAL	FICHA
LD L,(IX + d)	DD 6E XX	I-4
LD L, (IY + d)	FD 6E XX	I-4
LD L,A	6F	I-1
LD L,B	68	I-1
LD L,C	69	I-1
LD L,D	6A	I-1
LD L,E	6B	I-1
LD L,H	6C	I-1
LD L,L	6D	I-1
LD L,n	2E XX	I-1
LD R,A	ED 4F	I-2
LD SP, (nn)	ED 7B XX XX	I-9
LD SP,nn	31 XX XX	I-8
LD SP,HL	F9	I-11
LD SP,IX	DD F9	I-11
LD SP,IY	FD F9	I-11
LDD	ED A8	I-35
LDDR	ED B8	I-35
LDI	ED A0	I-34
LDIR	ED B0	I-34
NEG	ED 44	I-38
NOP	00	I-39
OR (HL)	B6	I-39
OR (IX + d)	DD B6 XX	I-23
OR (IY + d)	FD B6 XX	I-23
OR A	B7	I-23
OR B	B0	I-23
OR C	B1	I-23
OR D	B2	I-23
OR E	B3	I-23
OR H	B4	I-23
OR L	B5	I-23
OR n	F6 XX	I-23
OTDR	ED BB	I-67
OTIR	ED B3	I-66
OUT (C) ,A	ED 79	I-65
OUT (C) ,B	ED 41	I-65
OUT (C) ,C	ED 49	I-65
OUT (C) ,D	ED 51	I-65
OUT (C) ,E	ED 59	I-65
OUT (C) ,H	ED 61	I-65
OUT (C) ,L	ED 69	I-65
OUT (n) ,A	D3 XX	I-65
OUTD	ED AB	I-67
OUTI	ED A3	I-66
POP AF	F1	I-33
POP BC	C1	I-33
POP DE	D1	I-33
POP HL	E1	I-33
POP IX	DD E1	I-33
POP IY	FD E1	I-33
PUSH AF	F5	I-32
PUSH BC	C5	I-32
PUSH DE	D5	I-32
PUSH HL	E5	I-32
PUSH IX	DD E5	I-32
PUSH IY	FD E5	I-32
RES 0, (HL)	CB 86	I-54
RES 0, (IX + d)	DD CB XX 86	I-54
RES 0, (IY + d)	FD CB XX 86	I-54
RES 0,A	CB 87	I-53
RES 0,B	CB 80	I-53
RES 0,C	CB 81	I-53
RES O,D	CB 82	I-53
RES 0,E	CB 83	I-53
RES 0,H	CB 84	I-53
RES 0,L	CB 85	I-53
RES 1, (HL)	CB 8E	I-54
RES 1, (IX + d)	DD CB XX 8E	I-54
RES 1, (IY + d)	FD CB XX 8E	I-54
RES 1,A	CB 8F	I-53
RES 1,B	CB 8B	I-53
RES 1,C	CB 89	I-53
RES 1,D	CB 8A	I-53
RES 1,E	CB 8B	I-53
RES 1,H	CB 8C	I-53
RES 1,L	CB 8D	I-53
RES 1, (HL)	CB 96	I-54
RES 2, (IX + d)	DD CB XX 96	I-54
RES 2, (IY + d)	FD CB XX 96	I-54
RES 2,A	CB 97	I-53
RES 2,B	CB 90	I-53
RES 2,C	CB 91	I-53
RES 2,D	CB 92	I-53
RES 2,E	CB 93	I-53
RES 2,H	CB 94	I-53
RES 2,L	CB 95	I-53

MNEMONICO	HEXADECIMAL	FICHA
RES 3, (HL)	CB 9E	I-54
RES 3, (IX + d)	DD CB XX 9E	I-54
RES 3, (IY + d)	FD CB XX 9E	I-54
RES 3,A	CB 9F	I-51
RES 3,B	CB 98	I-53
RES 3,C	CB 99	I-53
RES 3,D	CB 9A	I-53
RES 3,E	CB 9B	I-53
RES 3,H	CB 9C	I-53
RES 3,L	CB 9D	I-53
RES 4, (HL)	CB A6	I-54
RES 4, (IX + d)	DD CB XX A6	I-54
RES 4, (IY + d)	FD CB XX A6	I-54
RES 4,A	CB A7	I-53
RES 4,B	CB A0	I-53
RES 4,C	CB A1	I-53
RES 4,D	CB A2	I-53
RES 4,E	CB A3	I-53
RES 4,H	CB A4	I-53
RES 4,L	CB A5	I-53
RES 5, (HL)	CB AE	I-54
RES 5, (IX + d)	DD CB XX AE	I-54
RES 5, (IY + d)	FD CB XX AE	I-54
RES 5,A	CB AF	I-53
RES 5,B	CB A8	I-53
RES 5,C	CB A9	I-53
RES 5,D	CB AA	I-53
RES 5,E	CB AB	I-53
RES 5,H	CB AC	I-53
RES 5,L	CB AD	I-53
RES 6, (HL)	CB B6	I-53
RES 6, (IX + d)	DD CB XX B6	I-54
RES 6, (IY + d)	FD CB XX B6	I-54
RES 6,A	CB B7	I-53
RES 6,B	CB B0	I-53
RES 6,C	CB B1	I-53
RES 6,D	CB B2	I-53
RES 6,E	CB B3	I-53
RES 6,H	CB B4	I-53
RES 6,L	CB B5	I-53
RES 7, (HL)	CB BE	I-54
RES 7, (IX + d)	DD CB XX BE	I-54
RES 7 (IY + d)	FD CB XX BE	I-54
RES 7,A	CB BF	I-53
RES 7,B	CB B8	I-53
RES 7,C	CB B9	I-53
RES 7,D	CB BA	I-53
RES 7,E	CB BB	I-53
RES 7,H	CB BC	I-53
RES 7,L	CB BD	I-53
RET	C9	I-60
RET C	D8	I-60
RET M	F8	I-60
RET NC	D0	I-60
RET NZ	C0	I-60
RET P	F0	I-60
RET PE	E8	I-60
RET PO	E0	I-60
RET Z	C8	I-60
RETI	ED 4D	I-61
RETN	ED 45	I-61
RL (HL)	CB 16	I-42
RL (IX + d)	DD CB XX 16	I-42
RL (IY + d)	FD CB XX 16	I-42
RL A	CB 17	I-42
RL B	CB 10	I-42
RL C	CB 11	I-42
RL D	CB 12	I-42
RL E	CB 13	I-42
RL H	CB 14	I-42
RL L	CB 15	I-42
RLA	17	I-42
RLC (HL)	CB 06	I-41
RLC (IX + d)	DD CB XX 06	I-41
RLC (IY + d)	FD CB XX 06	I-41
RLC A	CB 07	I-41
RLC B	CB 00	I-41
RLC C	CB 01	I-41
RLC D	CB 02	I-41
RLC E	CB 03	I-41
RLC H	CB 04	I-41
RLC L	CB 05	I-41
RLCA	07	I-41
RLD	ED 6F	I-48
RR (HL)	CB 1E	I-43
RR (IX + d)	DD CB XX 1E	I-43
RR (IY + d)	FD CB XX 1E	I-43

MNEMONICO	HEXADECIMAL	FICHA
RR A	CB 1F	I-43
RR B	CB 18	I-43
RR C	CB 19	I-43
RR D	CB 1A	I-43
RR E	CB 1B	I-43
RR H	CB 1C	I-43
RR L	CB 1D	I-43
RRA	1F	I-43
RRC (HL)	CB 0E	I-44
RRC (IX + d)	DD CB XX 0E	I-44
RRC (IY + d)	FD CB XX 0E	I-44
RRC A	CB 0F	I-44
RRC B	CB 08	I-44
RRC C	CB 09	I-44
RRC D	CB 0A	I-44
RRC E	CH 0B	I-44
RRC H	CB 0C	I-44
RRC L	CB 0D	I-44
RRCA	0F	I-44
RRD	ED 67	I-48
RST 00H	C7	I-61
RST 08H	CF	I-61
RST 10H	D7	I-61
RST 18H	DF	I-61
RST 20H	E7	I-61
RST 28H	EF	I-61
RST 30H	F7	I-61
RST 38H	FF	I-61
SBC A, (HL)	9E	I-21

MNEMONICO	HEXADECIMAL	FICHA
SBC A, (IX + d)	DD 9E XX	I-21
SBC A, (IY + d)	FD 9E XX	I-21
SBC A,A	9F	I-20
SBC A,B	98	I-20
SBC A,C	99	I-20
SBC A,D	9A	I-20
SBC A,E	9B	I-20
SBC A,H	9C	I-20
SBC A,L	9D	I-20
SBC A,n	DE XX	I-20
SBC HL,BC	ED 42	I-29
SBC HL,DE	ED 52	I-29
SBC HL,HL	ED 62	I-29
SBC HL,SP	ED 72	I-29
SCF	37	I-39
SET 0, (HL)	CB C6	I-52
SET 0, (IX + d)	DD CB XX C6	I-52
SET 0, (IY + d)	FD CB XX C6	I-52
SET 0,A	CB C7	I-51
SET 0,B	CB C0	I-51
SET 0,C	CB C1	I-51
SET 0,D	CB C2	I-51
SET 0,E	CB C3	I-51
SET 0,H	CB C4	I-51
SET 0,L	CB C5	I-51
SET 1, (HL)	CB CE	I-52
SET 1, (IX + d)	DD CB XX CE	I-52
SET 1, (IY + d)	FD CB XX CE	I-52
SET 1,A	CB CF	I-51

MNEMONICO	HEXADECIMAL	FICHA
SET 1,B	CB C8	I-51
SET 1,C	CB C9	I-51
SET 1,D	CB CA	I-51
SET 1,E	CB CB	I-51
SET 1,H	CB CC	I-51
SET 1,L	CB CD	I-51
SET 2, (HL)	CB D6	I-52
SET 2, (IX + d)	DD CB XX D6	I-52
SET 2, (IY + d)	FD CB XX D6	I-52
SET 2,A	CB D7	I-51
SET 2,B	CB D0	I-51
SET 2,C	CB D1	I-51
SET 2,D	CB D2	I-51
SET 2,E	CB D3	I-51
SET 2,H	CB D4	I-51
SET 2,L	CB D5	I-51
SET 3, (HL)	CB DE	I-52
SET 3, (IX + d)	DD CB XX DE	I-52
SET 3, (IY + d)	FD CB XX DE	I-52
SET 3,A	CB DF	I-51
SET 3,B	CB D8	I-51
SET 3,C	CB D9	I-51
SET 3,D	CB DA	I-51
SET 3,E	CB DB	I-51
SET 3,H	CB DC	I-51
SET 3,L	CB DD	I-51
SET 4, (HL)	CBE6	I-52
SET 4, (IX + d)	DD CB XX E6	I-52
SET 4, (IY + d)	FD CB XX E6	I-52

MNEMONICO	HEXADECIMAL	FICHA	MNEMONICO	HEXADECIMAL	FICHA	MNEMONICO	HEXADECIMAL	FICHA
SET 4,A	CB E7	I-51	SET 7,A	CB FF	I-51	SRL A	CB 3F	I-47
SET 4,B	CB E0	I-51	SET 7,B	CB F8	I-51	SRL B	CB 38	I-47
SET 4,C	CB E1	I-51	SET 7,C	CB F9	I-51	SRL C	CB 39	I-47
SET 4,D	CB E2	I-51	SET 7,D	CB FA	I-51	SRL D	CB 3A	I-47
SET 4,E	CB E3	I-51	SET 7,E	CB FB	I-51	SRL E	CB 3B	I-47
SET 4,H	CB E4	I-51	SET 7,H	CB FC	I-51	SRL H	CB 3C	I-47
SET 4,L	CB E5	I-51	SET 7,L	CB FD	I-51	SRL L	CB 3D	I-47
SET 5, (HL)	CB EE	I-52	SLA (HL)	CB 26	I-45	SUB (HL)	96	I-19
SET 5, (IX + d)	DD CB XX EE	I-52	SLA (IX + d)	DD CB XX 26	I-45	SUB (IX + d)	DD 96 XX	I-19
SET 5, (IY + d)	FD CB XX EE	I-52	SLA (IY + d)	FD CB XX 26	I-45	SUB (IY + d)	FD 96 XX	I-19
SET 5,A	CB EF	I-51	SLA A	CB 27	I-45	SUB A	97	I-18
SET 5,B	CB E8	I-51	SLA B	CB 20	I-45	SUB B	90	I-18
SET 5,C	CB E9	I-51	SLA C	CB 21	I-45	SUB C	91	I-18
SET 5,D	CB EA	I-51	SLA D	CB 22	I-45	SUB D	92	I-18
SET 5,E	CB EB	I-51	SLA E	CB 23	I-45	SUB E	93	I-18
SET 5,H	CB EC	I-51	SLA H	CB 24	I-45	SUB H	94	I-18
SET 5,L	CB ED	I-51	SLA L	CB 25	I-45	SUB L	95	I-18
SET 6, (HL)	CB F6	I-52	SRA (HL)	CB 2E	I-46	SUB n	D6 XX	I-18
SET 6, (IX + d)	DD CB XX F6	I-52	SRA (IX + d)	DD CB XX 2E	I-46	XOR (HL)	AE	I-24
SET 6, (IY + d)	FD CB XX F6	I-52	SRA (IY + d)	FD CB XX 2E	I-46	XOR (IX + d)	DD AE XX	I-24
SET 6,A	CB F7	I-51	SRA A	CB 2F	I-46	XOR (IY + d)	FD AE XX	I-24
SET 6,B	CB F0	I-51	SRA B	CB 28	I-46	XOR A	AF	I-24
SET 6,C	CB F1	I-51	SRA C	CB 29	I-46	XOR B	A8	I-24
SET 6,D	CB F2	I-51	SRA D	CB 2A	I-46	XOR C	A9	I-24
SET 6,E	CB F3	I-51	SRA E	CB 2B	I-46	XOR D	AA	I-24
SET 6,H	CB F4	I-51	SRA H	CB 2C	I-46	XOR E	AB	I-24
SET 6,L	CB F5	I-51	SRA L	CB 2D	I-46	XOR H	AC	I-24
SET 7, (HL)	CB FE	I-52	SRL (HL)	CB 3E	I-47	XOR L	AD	I-24
SET 7, (IX + d)	DD CB XX FE	I-52	SRL (IX + d)	DD CB XX 3E	I-47	XOR n	EE XX	I-24
SET 7, (IY + d)	FD CB XX FE	I-52	SRL (IY + d)	FD CB XX 3E	I-47			

Ficha	Instrucción	C	Z	P/V	S	N	H	Comentarios
I-20	ADC HL, ss	#	#	V	#	0	?	Suma de *16* bits con acarreo.
I-14-17	ADC s; ADD s	#	#	V	#	0	#	Suma de *8* bits sin o con acarreo.
I-28	ADD, DD, ss	#	—	—	—	0	?	Suma *16* bits.
I-22	AND s	0	#	P	#	0	1	«Y» lógico acumulador.
I-49-50	BIT b, m	—	#	?	?	0	1	Comprobación del estado de un bit.
I-39	CCF	#	—	—	—	0	?	Complementar el carry.
I-36-37	CPD; CPDR; CPI; CPIR	—	#	#	?	1	?	Instrucción de búsqueda de bloques Z = 1 si A = (HL) P/V = 0 si BC = 0.
I-25	CP s	#	#	V	#	1	#	Comparar acumulador.
I-38	CPL	—	—	—	—	1	1	Complementar acumulador.
I-38	DAA	#	#	P	#	—	#	Ajuste decimal acumulador.
I-27	DEC m	—	#	V	#	1	#	Decrementar *8* bits.
	IN r, (C)	—	#	P	#	0	0	Entrada direccionada por registro.
I-26	INC m	—	#	V	#	0	#	Incrementar *8* bits.
I-63-64	IND INI	—	#	?	?	1	?	Entrada de bloques Z = 1 si B = 0.
I-63-64	INDR;INIR	—	1	?	?	1	?	Entrada de bloques.

= indicador afectado; **—** = no afectado; **?** = desconocido; **P** = paridad; **V** = sobrepasamiento.

Ficha	Instrucción	C	Z	P/V	S	N	H	Comentarios
I-2	LD A,I; LD A,R	—	#	IFF2	#	0	0	El contenido del biestable de interrupciones se copia en P/V.
I-34-35	LDD; LDI	—	?	#	?	0	0	Instrucciones de transferencia de bloques.
I-34-35	LDDR; LDIR	—	?	0	?	0	0	P/V = 0 si BC = 0.
I-38	NEG	#	#	V	#	1	#	Negar acumulador.
I-23	OR s	0	#	P	#	0	0	«O» lógico acumulador.
I-66-67	OTDR; OTIR	—	1	?	?	1	?	Salida de bloques.
I-66-67	OUTD; OUTI	—	#	?	?	1	?	Salida de bloques Z = 1 si B = 0.
I-41-44	RLA; RLCA; RRA; RRCA	#	—	—	—	0	0	Rotación del acumulador.
I-48	RLD; RRD	—	#	P	#	0	0	Rotar dígitos izquierda y derecha.
I-41-44	RL m; RLC m; RR m; RRC m;	#	#	P	#	0	0	Rotar y desplazar bits.
I-45-47	SLA m; SRA m; SRL m							
I-29	SBC HL,ss	#	#	V	#	1	?	Restar *16* bits con acarreo.
I-39	SCF	1	—	—	—	0	0	Hacer carry = 1.
I-18-21	SBC s; SUB s			V		1		Restar *8* bits con acarreo.
I-24	XOR x	0		P		0	0	«O» exclusivo acumulador.

= indicador afectado; — = no afectado; ? = desconocido; **P** = paridad; **V** = sobrepasamiento.

Rutina	Dirección		Ficha	Rutina	Dirección		Ficha	Rutina	Dirección		Ficha
ADD-CHAR	0F81H	3969d	M-19	CHAN-S	1642H	5698d	M-23	COPY	0EACH	3756d	M-18
ALPHA	2C8DH	11405d	M-40	CHARS-T	3D00H	15616d	M-43	COPY-1	0EB2H	3762d	M-18
ALPHANUM	2C88H	11400d	M-40	CIRCLE	2320H	8992D	M-36	COPY-BUFF	0ECDH	3789d	M-18
AUTO-LIST	1795H	6037d	M-26	CIRCLE-1	232DH	9005d	M-36	COPY-LINE	0EF4H	3828d	M-18
BC-SPACES	0030H	48d	M-3	CL-ADDR	0E9BH	3739d	M-18	CP-LINES	1980H	6528d	M-27
BEEP	03F8H	1016d	M-8	CL-ALL	0DAFH	3503d	M-16	DATA	1E27H	7719d	M-31
BEEPER	03B5H	949d	M-8	CL-ATTR	0E88H	3720d	M-17	DE.(DE + 1)	2AEEH	10990d	M-39
BORDER	2294H	8852d	M-34	CL-LINE	0E44H	3652d	M-16	DEC-TO-FP	2C9BH	11419d	M-40
BREAK-KEY	1F54H	8020d	M-34	CL-SC-ALL	0DFEH	3582d	M-17	DEF-FN	1F60H	8032d	M-34
CA = 10A + C	2F88H	12171d	M-42	CL-SCROLL	0E00H	3584d	M-17	DIFFER	19DDH	6621d	M-28
CALCULATE	335BH	13147d	M-44	CL-SET	0DD9H	3545d	M-16	DIM	2C02H	11266d	M-40
CALL-JUMP	162CH	5676d	M-23	CLEAR	1EACH	7852d	M-32	DR3-PRMS1	2394H	9108d	M-36
CASS-MES	09A1H	2465d	M-11	CLEAR-PRB	0EDFH	3807d	M-18	DRAW	2382H	9090d	M-36
CAT-ETC	1793H	6035d	M-26	CLEAR-SP	1097H	4247d	M-19	DRAW-LINE	24B7H	9399d	M-36
CH-ADD + 1	0074H	116d	M-5	CLOSE	16E5H	5861d	M-24	DRAW-LINE-1	24BAH	9402d	M-36
CHAN-FLAG	1615H	5653d	M-23	CLS	0D6BH	3435d	M-16	E-LINE-NO	19FBH	6651d	M-28
CHAN-K	1634H	5684d	M-23	CO-TEMP	21E1H	8673d	M-34	EACH-STMT	198BH	6539d	M-27
CHAN-OPEN	1601H	5633d	M-23	CONT-CHAR	0A11H	2577d	M-12	ED-COPY	111DH	4381d	M-20
CHAN-P	164DH	5709d	M-23	CONTINUE	1E5FH	7775d	M-32	ED-DELETE	1015H	4117d	M-19

Rutina	Dirección		Ficha	Rutina	Dirección		Ficha	Rutina	Dirección		Ficha
ED-DOWN	0FF3H	4083d	M-19	FOR	1D03H	7427d	M-31	INT-STORE	2D8EH	11662d	M-41
ED-EDGE	1031H	4145d	M-19	FP-CALC	0028H	40d	M-3	INT-TO-FP	2D3BH	11579d	M-41
ED-EDIT	0FA9H	4009d	M-19	FP-DELETE	2DADH	11693d	M-42	K-DECODE	0333H	819d	M-7
ED-ENTER	1031H	4145d	M-19	FP-TO-A	2DD5H	11733d	M-42	KEY-INPUT	10A8H	4264d	M-20
ED-ERROR	107FH	4223d	M-19	FP-TO-BC	2DA2H	11682d	M-42	KEY-SCAN	028EH	654d	M-6
ED-GRAPH	107CH	4220d	M-19	FREE-MEM	1F1AH	7962d	M-33	KEY-TABLES	0205H	517d	M-5
ED-IGNORE	101EH	4126d	M-19	GET-CHAR	0018H	24d	M-2	KEYBOARD	02BFH	703d	M-6
ED-KEYS	0E92H	3986d	M-19	GO-TO	1E67H	7783d	M-32	L-ENTER	2BA6H	11174d	M-39
ED-LEFT	1007H	4103d	M-19	GOSUB	1EEEH	1917d	M-33	LD-BLOCK	0802H	2050d	M-10
ED-RIGHT	100CH	4108d	M-19	HL = HL*DE	2DA9H	12457d	M-42	LD-BYTES	0556H	1366d	M-10
ED-SYMBOL	1076H	4214d	M-19	IF	1CF0H	7408d	M-31	LD-CONTRL	0808H	2056d	M-10
ED-UP	1059H	4185d	M-19	IN-CHAN-K	21D6H	8662d	M-34	LD-EDGE1	05E7H	1511d	M-10
EDITOR	0F2CH	3884d	M-19	INDEXER	16DCH	5882d	M-25	LD-EDGE2	05E3H	1507d	M-10
ERROR-1	0008H	8d	M-1	INIT-CHAN	15AFH	5551d	M-22	LET	2AFFH	11007d	M-39
EXPT-1NUM	1C82H	7298d	M-30	INIT-STRM	15C6H	5574d	M-22	LINE-ADDR	196EH	6510d	M-27
EXPT-2NUM	1C7AH	7290d	M-30	INPUT	2089H	8329d	M-34	LINE-DRAW	2477H	9335d	M-36
FETCH-NUM	1CDEH	7390d	M-30	INPUT-AD	15E6H	5606d	M-22	LINE-NO	1695H	5781d	M-25
FIND-INT-1	1E94H	7828d	M-32	INT-EXP	2ACCH	10956D	M-39	LINE-RUN	1B8AH	7050d	M-29
FIND-INT-2	1E99H	7833d	M-32	INT-FETCH	2D7FH	11647d	M-41	LINE-SCAN	1B17H	6935d	M-29

Rutina	Dirección		Ficha
LIST	17F9H	6137d	M-26
LIST-ALL	1835H	6197d	M-26
LLIST	17F5H	6133d	M-26
LN-FETCH	190FH	6415d	M-27
LOOK-PROG	1D86H	7558d	M-31
LOOK-VARS	24FBH	10418d	M-38
LPRINT	1FC9H	8137d	M-34
MAIN-1	12A9H	4777d	M-21
MAIN-2	12ACH	4780d	M-21
MAIN-3	12CFH	4815d	M-21
MAIN-4	1303H	4867d	M-21
MAIN-5a9	133CH	4924d	M-21
MAIN-ADD	155DH	5469d	M-21
MAIN-EXEC	12A2H	4770d	M-21
MAKE-ROOM	1655H	5717d	M-24
MASK-INT	0038H	56d	M-4
ME-CONTRL	08B6H	2230d	M-11
ME-ENTER	092CH	2348d	M-11
NEW	1187H	4535d	M-21
NEXT-CHAR	0020H	32d	M-2
NEXT-ONE	19B8H	6584d	M-27

Rutina	Dirección		Ficha
NEXT	1DABH	7595d	M-31
NUMERIC	2D1BH	11547d	M-40
ONE-SPACE	1652H	5714d	M-24
OPEN	1736H	5942d	M-26
OUT	1E7AH	7802d	M-32
OUT-CODE	15EFH	5615d	M-23
OUT-LINE	1855H	6229d	M-26
OUT-NUM-1	1A1BH	6683d	M-28
OUT-NUM-2	1A28H	6696d	M-28
P-INT-STO	2D8CH	11660d	M-41
PAUSE	1F3AH	7994d	M-33
PAUSE-1	1F3DH	7997d	M-33
PERMS	1C96H	7318d	M-30
PIXEL-ADD	22AAH	8874d	M-35
PLOT	22DCH	8924d	M-35
PLOT-BC	22DFH	8927d	M-35
PO-ABLE	0AD9H	2777d	M-12
PO-ANY	0B24H	2852d	M-13
PO-ATTR	0BDBH	3035d	M-14
PO-BACK1	0A23H	2595d	M-12
PO-CHANGE	0A80H	2688d	M-12

Rutina	Dirección		Ficha
PO-CHAR	0B65H	2917d	M-14
PO-COMMA	0A5FH	2655d	M-12
PO-CONT	0A87H	2695d	M-12
PO-ENTER	0A4FH	2639d	M-12
PO-FETCH	0B03H	2819d	M-13
PO-GR-1	0B38H	2872d	M-13
PO-MSG	0C0AH	3082d	M-15
PO-QUEST	0A69H	2665d	M-12
PO-RIGHT	0A3DH	2521d	M-12
PO-SAVE	0C3BH	3131d	M-15
PO-SCR	0C55H	3157d	M-17
PO-SEARCH	0C41H	3137d	M-15
PO-STORE	0ADCH	2780d	M-13
PO-T&UDG	0B52H	2898d	M-13
PO-TABLE	0C14H	3092d	M-15
PO-TOKENS	0C10H	3088d	M-15
PO-TV-2	0A6DH	2669d	M-12
POINT-BC	22CEH	8910d	M-35
POINT-SUB	22CBH	8907d	M-35
POINTERS	1664H	5732d	M-24
POKE	1E80H	7808d	M-32

Rutina	Dirección		Ficha
PR-ALL	0B7FH	2943d	M-14
PRINT	1FCDH	8141d	M-34
PRINT-2	1FDFH	8159d	M-34
PRINT-A-1	0010H	16d	M-2
PRINT-A-2	15F2H	5618d	M-23
PRiNT-FP	2DE3H	11747d	M-42
PRINT-OUT	09F4H	2548d	M-12
RANDOMIZE	1E4FH	7759d	M-31
READ	1DECH	7660d	M-31
RECLAIM-1	19E5H	6629d	M-28
RECLAIM-2	19E8H	6632d	M-28
REM	1BB2H	7090d	M-30
REMOVE-FP	11A7H	4519d	M-20
REP-MESS	1391H	5009d	M-21
REPORT-G	1555H	5461d	M-21
RESERVE	169EH	5779d	M-25
RESET	0066H	102d	M-4
RESTORE	1E42H	7746d	M-31
RETURN	1F23H	7971d	M-33
RUN	1EA1H	7841d	M-32

Rutina	Dirección		Ficha
S-ATTR-S	2580H	9600d	M-37
S-SCRN$-S	2535H	9525d	M-37
S-SCRNS-1	253FH	9535d	M-37
SA-BYTES	04C2H	1218d	M-9
SA-CONTRL	0970H	2416d	M-9
SA/LD-RET	053FH	1343d	M-9
SAVE-ETC	0605H	1541d	M-9
SCANNING	24FBH	9467d	M-37
SET-DE	1195H	4501d	M-20
SET-HL	1190H	4496d	M-20
SET-MIN	16B0H	5808d	M-25
SET-STK	16C5H	5829d	M-25
SET-WORK	16BFH	5823d	M-25
SKIP-OVER	007DH	125d	M-5
SLICING	2A52H	10834d	M-38
SP-SPACE	386EH	14446d	M-43
STACK-A	2D28H	11560d	M-40
STACK-BC	202BH	11563d	M-40
STACK-NUM	33B4H	13236d	M-43
START	0000H	0d	M-1

Rutina	Dirección		Ficha
START/NEW	11CBH	4555d	M-21
STK-DIGIT	2D22H	11554d	M-40
STK-FETCH	2BF1H	11249d	M-39
STK-PNTRS	35BFH	13759d	M-43
STK-STORE	24FBH	10934d	M-39
STK-TO-BC	2307H	8967d	M-35
STK-VAR	2996H	10646d	M-38
STOP	1CEEH	7406d	M-31
SWAP-BYTE	343EH	13374d	M-43
TEMP-PTR-1	0077H	119d	M-5
TEMPS	0D4DH	3405d	M-15
TEST-ROOM	1F05H	7941d	M-33
TEST-ZERO	34E9H	13545d	M-43
TOKEN-TABLE	0095H	149d	M-5
TWO-PARAM	1E85H	7813d	M-32
UNSTACK-Z	1FC3H	8131d	M-34
VAL-FET	1C56H	7254d	M-30
VAR-A-1	1C22H	7002d	M-30
VR-CONTRL	07CBH	1995	M-11
WAIT-KEY	15D4H	5588d	M-22

Código	Operación	Dirección	Ficha
0 00H	jump-true	368FH	M-50
1 01H	exchange	343CH	M-49
2 02H	delete	33A1H	M-49
3 03H	subtract	30OFH	M-45
4 04H	multiply	30CAH	M-45
5 05H	division	31AFH	M-45
6 06H	to-power	3851H	M-46
7 07H	or	351BH	M-48
8 08H	no-&-no	3524H	M-48
9 09H	no-l-eql	353BH	M-48
10 0AH	no-gr-eq	353BH	M-48
11 0BH	nos-neql	353BH	M-48
12 0CH	no-grtr	353BH	M-48
13 0DH	no-less	353BH	M-48
14 0EH	nos-eql	353BH	M-48
15 0FH	addition	3014H	M-45
16 10H	str-&-no	352DH	M-48
17 11H	str-l-eql	353BH	M-48
18 12H	str-gr-eq	353BH	M-48
19 13H	strs-neql	353BH	M-48

Código	Operación	Dirección	Ficha
20 14H	str-grtr	353BH	M-48
21 15H	str-less	353BH	M-48
22 16H	strs-eql	353BH	M-48
23 17H	strs-add	359CH	M-47
24 18H	val$	35DEH	M-47
25 19H	usr-$	34BCH	M-47
26 1AH	read-in	3645H	M-49
27 1BH	negate	346EH	M-46
28 1CH	code	3669H	M-47
29 1DH	val	35DEH	M-47
30 1EH	len	3674H	M-47
31 1FH	sin	37B5H	M-45
32 20H	cos	37AAH	M-45
33 21H	tan	37DAH	M-45
34 22H	asn	3833H	M-45
35 23H	acs	3843H	M-45
36 24H	atn	37E2H	M-45
37 25H	ln	3713H	M-46
38 26H	exp	36C4H	M-46
39 27H	int	36AFH	M-46

Código	Operación	Dirección	Ficha
40 28H	sqr	384AH	M-46
41 29H	sgn	3492H	M-48
42 2AH	abs	346AH	M-46
43 2BH	peek	34ACH	M-46
44 2CH	in	34A5H	M-46
45 2DH	usr-no	34B3H	M-46
46 2EH	str$	361FH	M-47
47 2FH	chr$	35C9H	M-47
48 30H	not	3501H	M-48
49 31H	duplicate	33COH	M-49
50 32H	n-mod-m	36A0H	M-49
51 33H	jump	3686H	M-50
52 34H	stk-data	33C6H	M-51
53 35H	dec-jr-nz	367AH	M-50
54 36H	less-0	3506H	M-48
55 37H	greater-0	34F9H	M-48
56 38H	end-calc	369BH	M-45
57 39H	get-argt	3783H	M-45
58 3AH	truncate	3214H	M-46
59 3BH	fp-calc-2	33A2H	M-45
60 3CH	e-to-fp	2D4FH	M-49

Código	Operación	Dirección	Ficha
61 3DH	re-stack	3297H	M-49
134 86H	series-06	3449H	M-51
136 88H	series-08	3449H	M-51
140 8CH	series-0C	3449H	M-51
160 A0H	stk-zero	341BH	M-50
161 A1H	stk-one	341BH	M-50
162 A2H	stk-half	341BH	M-50
163 A3H	stk-pi/2	341BH	M-50
164 A4H	stk-ten	341BH	M-50
192 C0H	stk-mem-0	342DH	M-51
193 C1H	stk-mem-1	342DH	M-51
194 C2H	stk-mem-2	342DH	M-51
195 C3H	stk-men-3	342DH	M-51
196 C4H	stk-mem-4	342DH	M-51
197 C5H	stk-mem-5	342DH	M-51
224 E0H	get-mem-0	340FH	M-51
225 E1H	get-mem-1	340FH	M-51
226 E2H	get-mem-2	340FH	M-51
227 E3H	get-mem-3	340FH	M-51
228 E4H	get-mem-4	340FH	M-51
229 E5H	get-mem-5	340FH	M-51

En cada ficha se estudian los mnemónicos genéricos de cada microinstrucción de la CPU Z80A, operandos incluidos, con la descripción de lo que es cada operación y su codificación binaria (código de máquina), hexadecimal y decimal.

Se conocen además los ciclos de máquina, y los estados de cada ciclo, que usaremos para calcular el tiempo de ejecución de las operaciones, simplemente multiplicando el número total de estados por 0.3 us (millonésimas de segundo), teniendo en cuenta que el resultado es aproximado, debido a la estructura del Hardware del ZX Spectrum.

También se relacionan los indicadores afectados, que usaremos para las posteriores operaciones condicionales.

En las operaciones genéricas que tienen varias codificaciones posibles, según los operandos utilizados, se aplicarán las siguientes tablas de codificación parcial:

Mnemónico
Operando
Codificación
Tiempo de ejecución
Indicadores de condición
Grupos operacionales

r o r'

cualquiera de los registros de 8 bits:

A	111
B	000
C	001
D	010
E	011
H	100
L	101

s

cualquier posición de 8 bits:

- r
- n
- (HL)
- (IX+d)
- (IY+d)

dd o ss

cualquiera de los pares de registros:

BC 00
DE 01
HL 10
SP 11

qq

cualquera de los pares de registros:

BC 00
DE 01
HL 10
AF 11

pp

cualquiera de los pares de registros:

BC 00
DE 01
IX 10
SP 11

rr

cualquiera de los pares de registros:

BC 00
DE 01
IY 10
SP 11

---

cc

comprobar condición:

000 NZ (no cero)
001 Z (cero)
010 NC (no acarreo)
011 C (acarreo)
100 PO (paridad par)
101 PE (paridad impar)
110 P (positivo)
111 M (negativo)

b

comprobar bit:

000 0
001 1
010 2
011 3
100 4
101 5
110 6
111 7

t

direcciones de RESTART:

t	p
000	0000H
001	0008H
010	0010H
011	0018H
100	0020H
101	0028H
110	0030H
111	0038H

d

desplazamiento de 8 bits, en complemento a 2, rango de –128 a 127, ha de sumarse a la dirección actual.

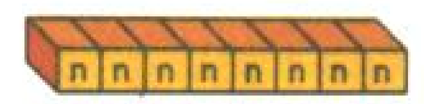
LD r,n

El número n de 8 bits es transferido a cualquier registro r.
Mnemónico: LD **Operandos:** r, n

Formato Binario:



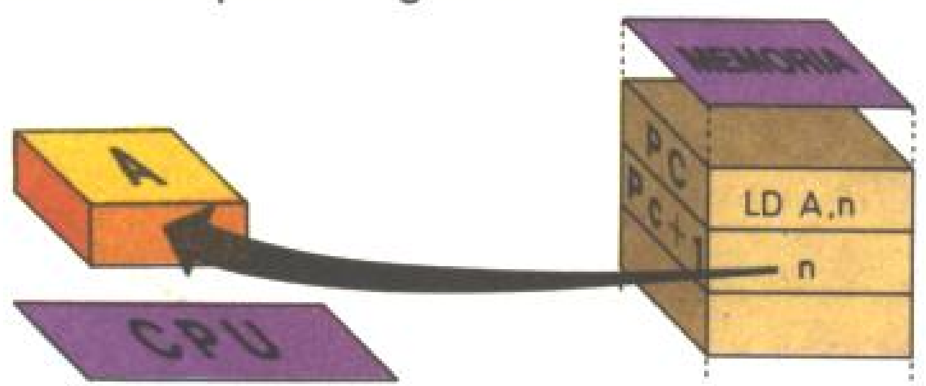
Ciclos: 2
Estados: 7 (4+3)



Indicadores: ninguno.

Ejemplo:

Si el registro A contiene 97H, después de ejecutar la instrucción
LD A,33H (binario 00111110,00110011)
resultará que el registro A contiene 33H.



Instr.	Hex.	Dec.	Instr.	Hex.	Dec.
LD A,A	7F	127	LD D,E	53	83
LD A,B	78	120	LD D,H	54	84
LD A,C	79	121	LD D,L	55	85
LD A,D	7A	122	LD D,n	16,n	22,n
LD A,E	7B	123	LD E,A	5F	95
LD A,H	7C	124	LD E,B	58	88
LD A,L	7D	125	LD E,C	59	89
LD A,n	3E,n	62,n	LD E,D	5A	90
LD B,A	47	71	LD E,E	5B	91
LD B,B	40	64	LD E,H	5C	92
LD B,C	41	65	LD E,L	5D	93
LD B,D	42	66	LD E,n	1E,n	30,n
LD B,E	43	67	LD H,A	67	103
LD B,H	44	68	LD H,B	60	96
LD B,L	45	69	LD H,C	61	97
LD B,n	06,n	6,n	LD H,D	62	98
LD C,A	4F	79	LD H,E	63	99
LD C,B	48	72	LD H,H	64	100
LD C,C	49	73	LD H,L	65	101
LD C,D	4A	74	LD H,n	26,n	38,n
LD C,E	4B	75	LD L,A	6F	111
LD C,H	4C	76	LD L,B	68	104
LD C,L	4D	77	LD L,C	69	105
LD C,n	0E,n	14,n	LD L,D	6A	106
LD D,A	57	87	LD L,E	6B	107
LD D,B	50	80	LD L,H	6C	108
LD D,C	51	81	LD L,L	6D	109
LD D,D	52	82	LD L,n	2E,n	46,n

LD r, r'

El contenido de cualquier registro r' es transferido a cualquier registro r.

Mnemónico: LD **Operandos:** r, r'

Formato binario:



Ciclos: 1
Estados: 4

Indicadores: ninguno

Registros r y r'

A = 111
B = 000
C = 001
D = 010
E = 011
H = 100
L = 101

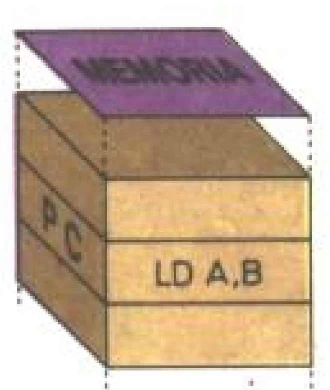
Ejemplo: LD A C



Ejemplo:

Si el registro B contiene 7AH, y el registro A contiene D4H, después de ejecutar la instrucción LD A,B (Binario 01111000)
resultará que ambos registros A y B contienen 7AH, valor que contenía el registro de origen (source), en este caso B.





LD R,A

El contenido del registro A es transferido al registro R.

Mnemónico: LD **Operandos:** R, A

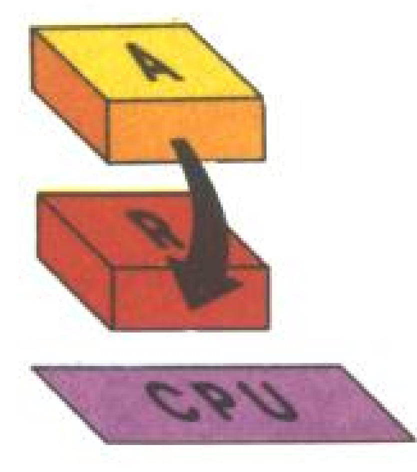
Formato binario:

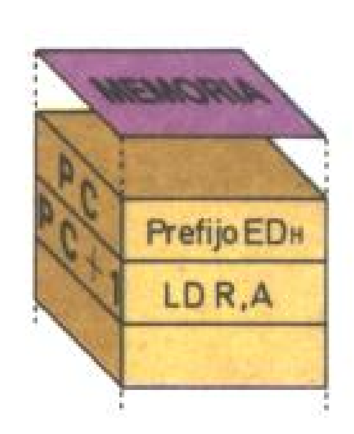


0 1 0 0 1 1 1 1

Ciclos: 2
Estados: 9 (4,5)

Indicadores: ninguno





Instr.	Hex.	Dec.
LD A,I	ED,57	237,87
LD I,A	ED,47	237,71
LD A,R	ED,5F	237,95
LD R,A	ED,4F	237,79

LD A,R

El contenido del registro R es transferido al registro A.

Mnemónico: LD **Operandos:** A,R

Formato binario:



0 1 0 1 1 1 1 1

Ciclos: 2
Estados: 9 (4,5)

Indicadores:

S – a 1 si R es negativo
Z – a 1 si R es 0
H – a 0
P/V – contenido de IFF2
N – a 0

LD I,A

El contenido del registro A es transferido al registro I.

Mnemónico: LD **Operandos:** I,A

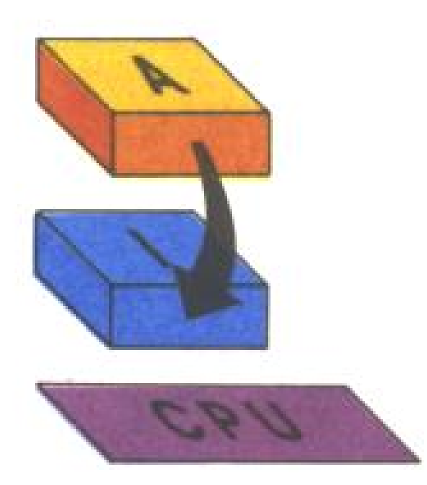
Formato binario:

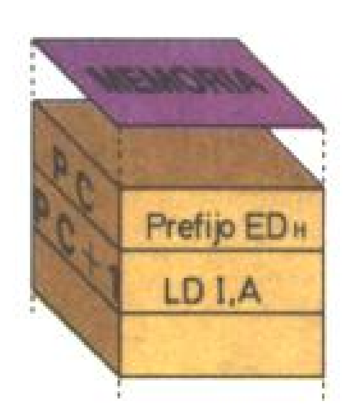
1 1 1 0 1 1 0 1

0 1 0 0 0 1 1 1

Ciclos: 2
Estados: 9 (4,5)

Indicadores: ninguno





LD A,I

El contenido del registro I es transferido al registro A.

Mnemónico: LD **Operandos:** A,I

Formato binario:

1 1 1 0 1 1 0 1



Ciclos: 2
Estados: 9 (4,5)

Indicadores:
S – a 1 si I es negativo
Z – a 1 si I es 0
H – a 0
P/V – contenido de IFF2
N – a 0

Ejemplo:

Si el registro I contiene 37H, después de ejecutar la instrucción
LD A,I
resultará que el registro A contiene 37 H, y los indicadores S y Z están a 0.

LD A,(nn) LD (nn),A

LD A,(nn)

El contenido de cualquier dirección de memoria especificada por el operando nn es transferido al registro A.

Mnemónico: LD **Operandos:** A,(nn)

Formato binario:



n n n n n n n n

n n n n n n n n

Ciclos: 4
Estados: 13 (4,3,3,3)

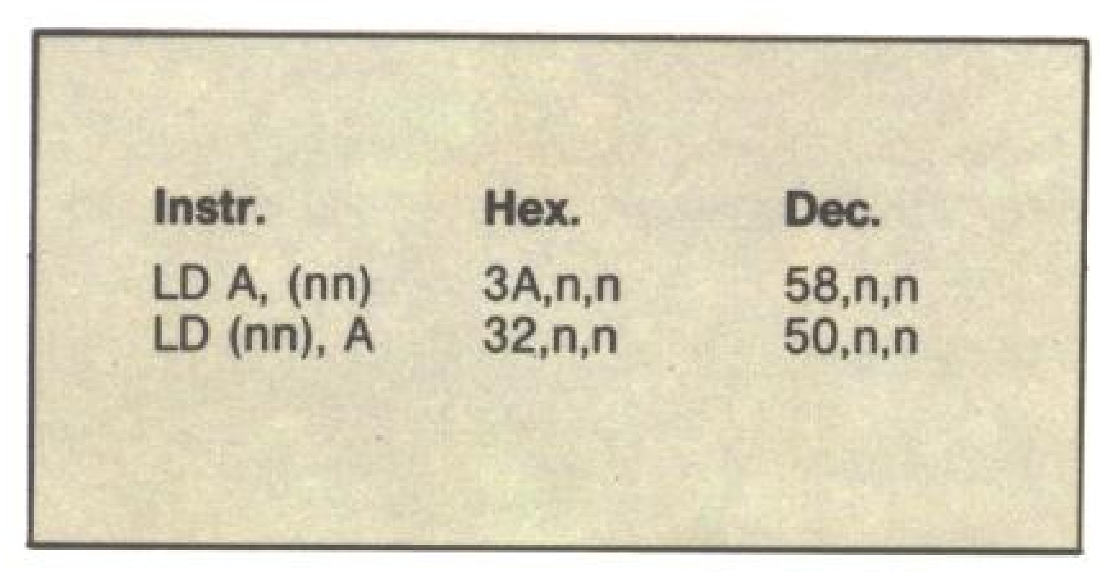
Indicadores: ninguno

Ejemplo:

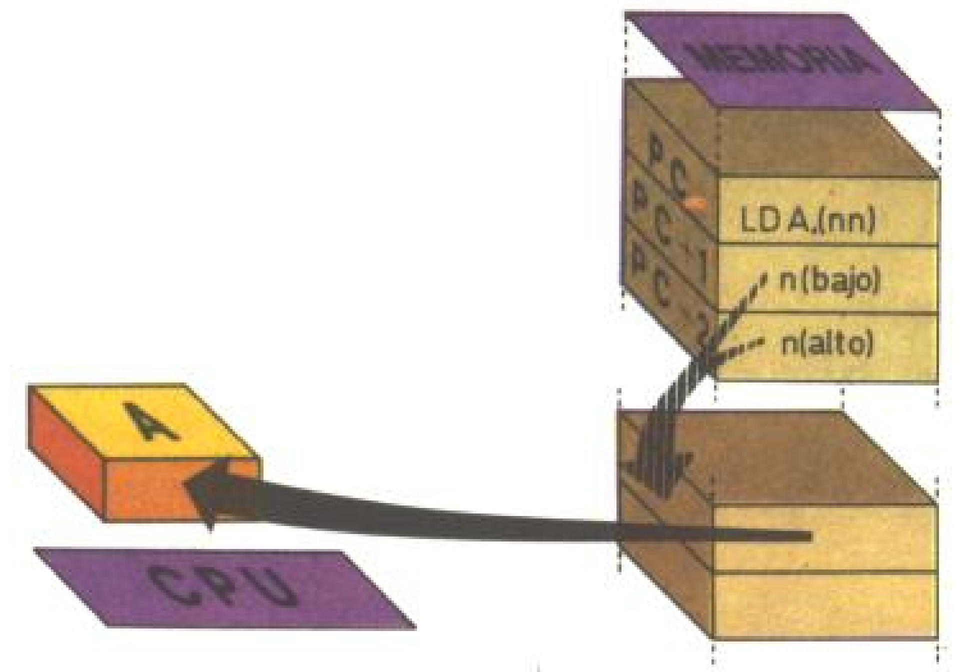
Si el contenido de la dirección de memoria 5AF0H es 07H, después de ejecutar la instrucción

LD A, (5AF0H)

resultará que el registro A contiene 07H.



Instr.	Hex.	Dec.
LD A, (nn)	3A,n,n	58,n,n
LD (nn), A	32,n,n	50,n,n



LD (nn),A

El contenido del registro A es transferido a la dirección de memoria especificada por el operando nn.

Mnemónico: LD **Operandos:** (nn),A

Formato binario:

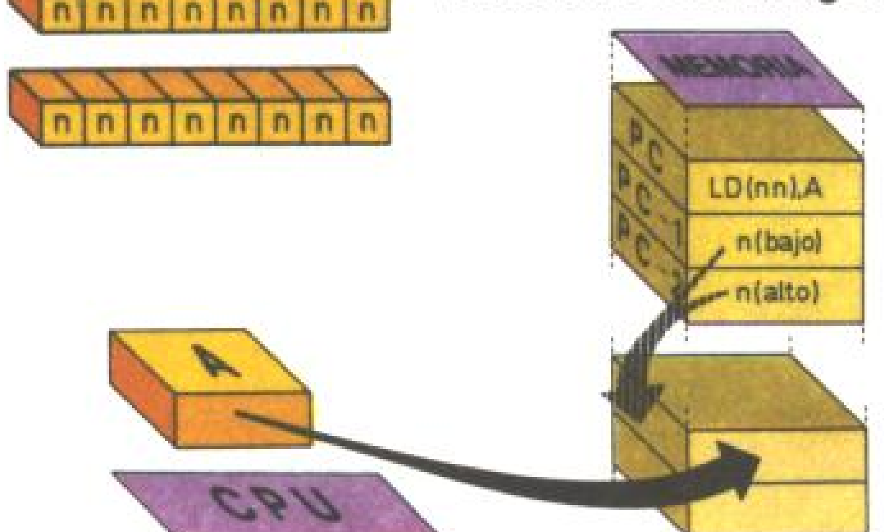
0 0 1 1 0 0 1 0

n n n n n n n n

n n n n n n n n

Ciclos: 4
Estados: 13 (4,3,3,3)

Indicadores: ninguno



Ejemplo:

Si el contenido del registro A es 90H, después de ejecutar la instrucción

LD (4000H),A

resultará que la dirección de memoria 4000H contiene 90H.

• Estas instrucciones equivalen a las correspondientes LD A,(HL) y LD (HL),A, cuando se trata de transferir un solo número de 8 bits entre el registro A y la dirección de memoria especificada. El ejemplo quedaría de la forma

LD HL,4000H
LD (HL),A

ofreciendo la ventaja de que al utilizar una instrucción en lugar de dos, la subrutina ocupa menos memoria, y es más rápida de ejecución.

LD (HL),n

El número n de 8 bits es transferido a la dirección de memoria especificada por el contenido del par HL.

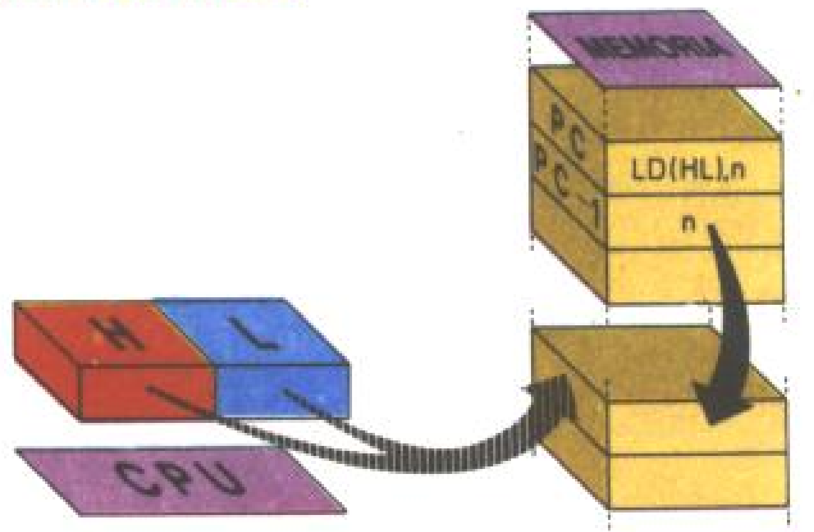
Mnemónico: LD **Operandos:** (HL),n

Formato binario:

0	0	1	1	0	1	1	0
n	n	n	n	n	n	n	n

Ciclos: 3
Estados: 1Ø (4,3,3)

Indicadores: ninguno



Instr.	Hex.	Dec.
LD (HL),n	36,n	54,n
LD (HL),A	77	119
LD (HL),B	70	112
LD (HL),C	71	113
LD (HL),D	72	114
LD (HL),E	73	115
LD (HL),H	74	116
LD (HL),L	75	117
LD A,(HL)	7E	126
LD B,(HL)	46	70
LD C,(HL)	4E	78
LD D,(HL)	56	86
LD E,(HL)	5E	94
LD H,(HL)	66	102
LD L,(HL)	6E	110

LD (HL),r

El contenido del registro r es transferido a la dirección de memoria especificada por el contenido del par HL.

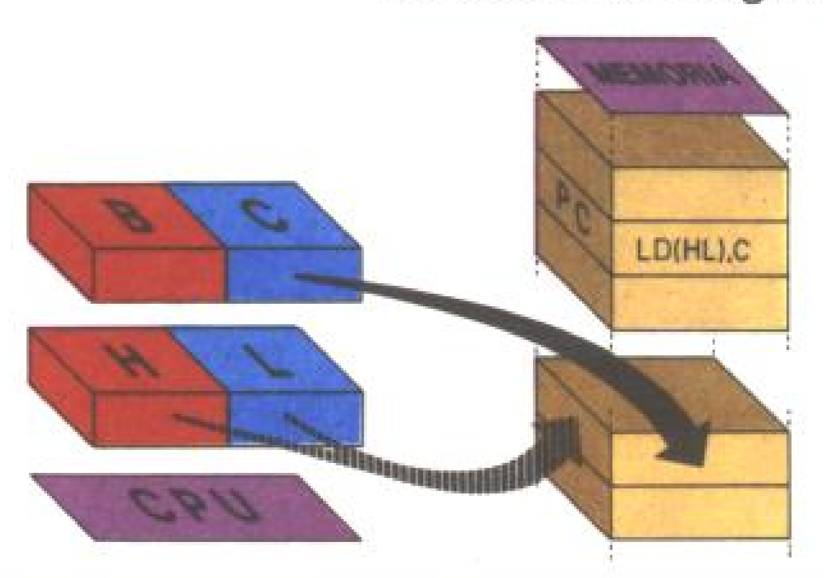
Mnemónico: LD **Operandos:** (HL),r

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

Indicadores: ninguno

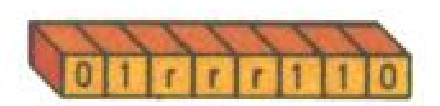


LD r,(HL)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del par HL es transferido al registro r.

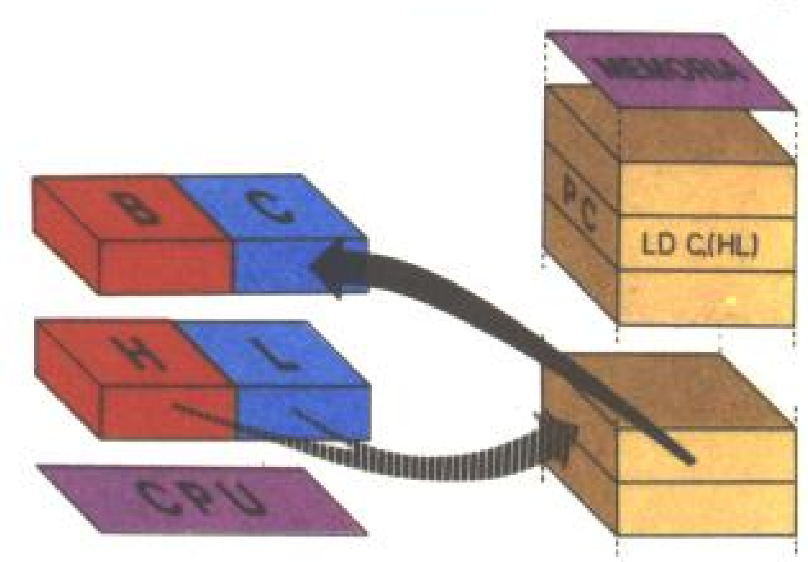
Mnemónico: LD **Operandos:** r,(HL)

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

Indicadores: ninguno

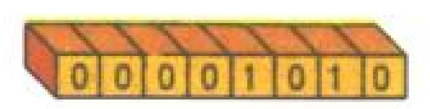


LD A,(BC)

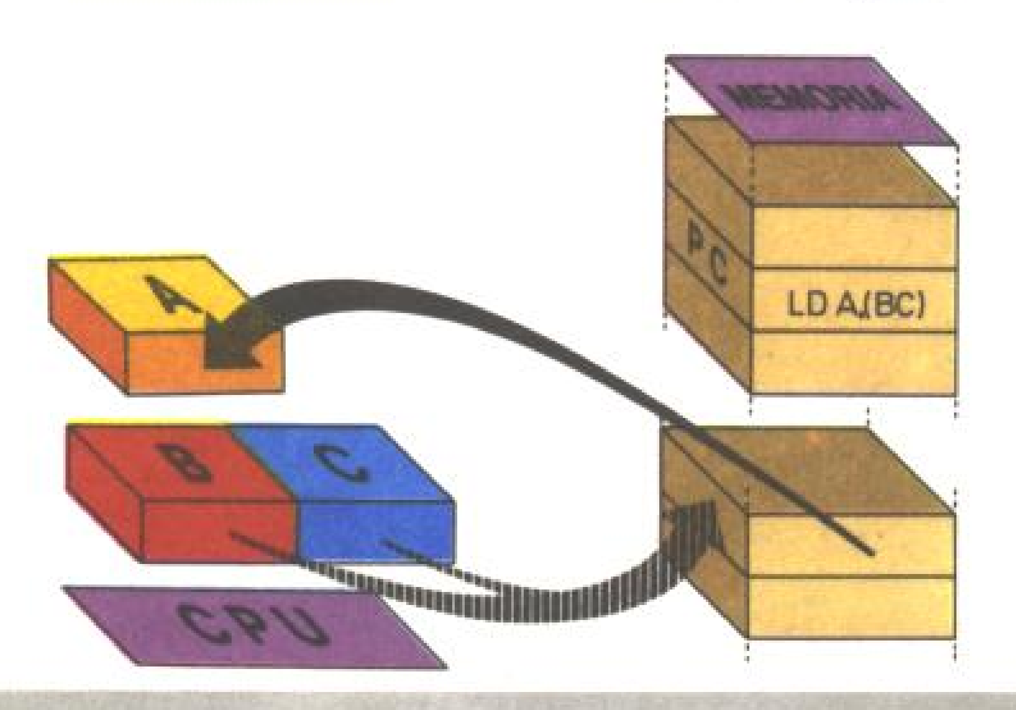
El contenido de 8 bits de la dirección de memoria especificada por el contenido del par BC es transferido al registro A.

Mnemónico: LD **Operandos:** A,(BC)

Formato binario:



Ciclos: 2
Estados: 7 (4,3)
Indicadores: ninguno



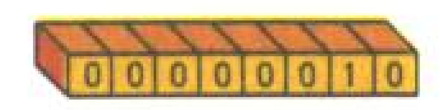
Instr.	Hex.	Dec.	Instr.	Hex.	Dec.
LD A,(BC)	0A	10	LD (BC),A	02	2
LD A, (DE)	1A	26	LD (DE),A	12	18

LD (BC),A

El contenido del registro A es transferido a la dirección de memoria especificada por el contenido del par BC.

Mnemónico: LD **Operandos:** (BC),A

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

Indicadores: ninguno

Ejemplo:

Si el contenido del par BC es 3000H, y el contenido del registro A es 7FH, después de ejecutar la instrucción: LD (BC),A
resultará que la dirección de memoria 3000H contiene 7FH.

LD A,(DE)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del para DE es transferido al registro A.

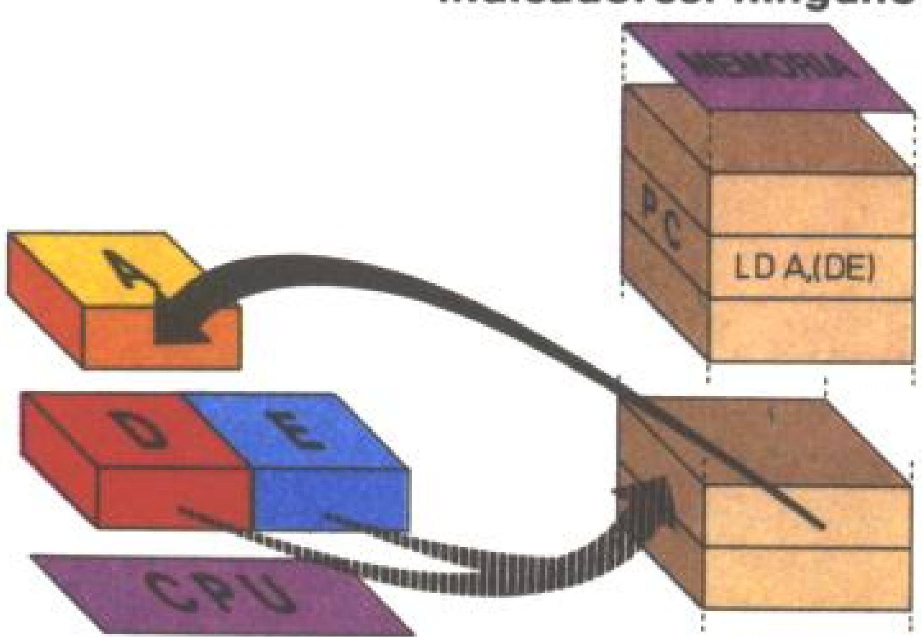
Mnemónico: LD **Operandos:** A,(DE)

Formato binario:

| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ninguno



LD (DE),A

El contenido del registro A es transferido a la dirección de memoria especificada por el contenido del par DE.

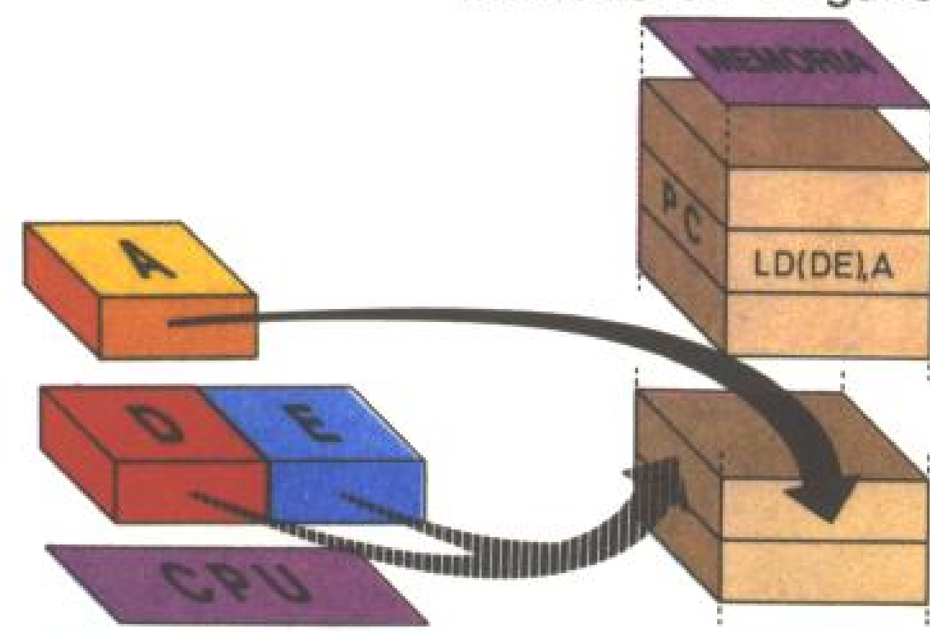
Mnemónico: LD **Operandos:** (DE),A

Formato binario:

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ninguno



LD (IX+d),n LD (IX+d),r LD r,(IX+d)

LD (IX+d),n

El número de 8 bits n es transferido a la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d (d es un número de 8 bits en complemento a 2).

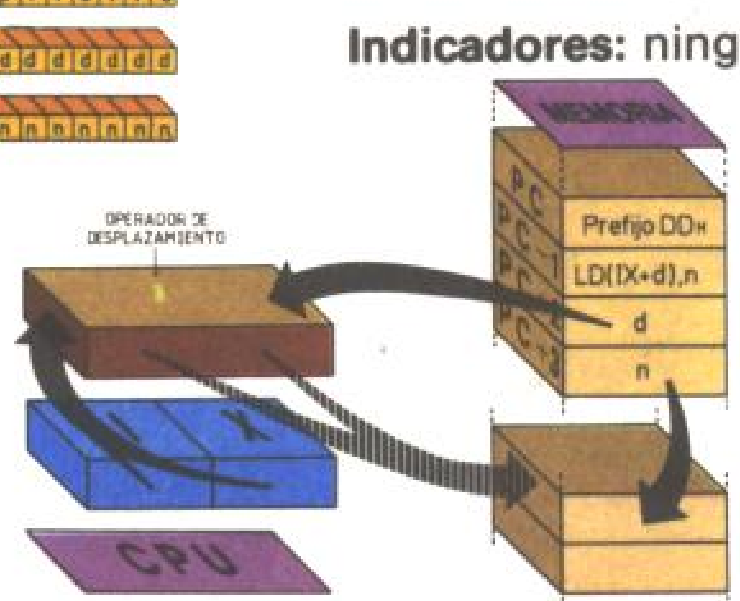
Mnemónico: LD **Operandos:** (IX+d),n

Formato binario:

1 1 0 1 1 1 0 1
0 0 1 1 0 1 1 0
d d d d d d d d
n n n n n n n n

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



Instr.	Hex.	Dec.
LD (IX+d),n	DD,36,d,n	221,54,d,n
LD (IX+d),A	DD,77,d	221,119,d
LD (IX+d),B	DD,70,d	221,112,d
LD (IX+d),C	DD,71,d	221,113,d
LD (IX+d),D	DD,72,d	221,114,d
LD (IX+d),E	DD,73,d	221,115,d
LD (IX+d),H	DD,74,d	221,116,d
LD (IX+d),L	DD,75,d	221,117,d
LD A,(IX+d)	DD,7E,d	221,126,d
LD B,(IX+d)	DD,46,d	221,70,d
LD C,(IX+d)	DD,4E,d	221,78,d
LD D,(IX+d)	DD,56,d	221,86,d
LD E,(IX+d)	DD,5E,d	221,94,d
LD H,(IX+d)	DD,66,d	221,102,d
LD L,(IX+d)	DD,6E,d	221,110,d

LD (IX+d),r

El contenido de cualquier registro r es transferido a la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d (d es un número de 8 bits en complemento a 2).

Mnemónico: LD **Operandos:** (IX+d),r

Formato binario:

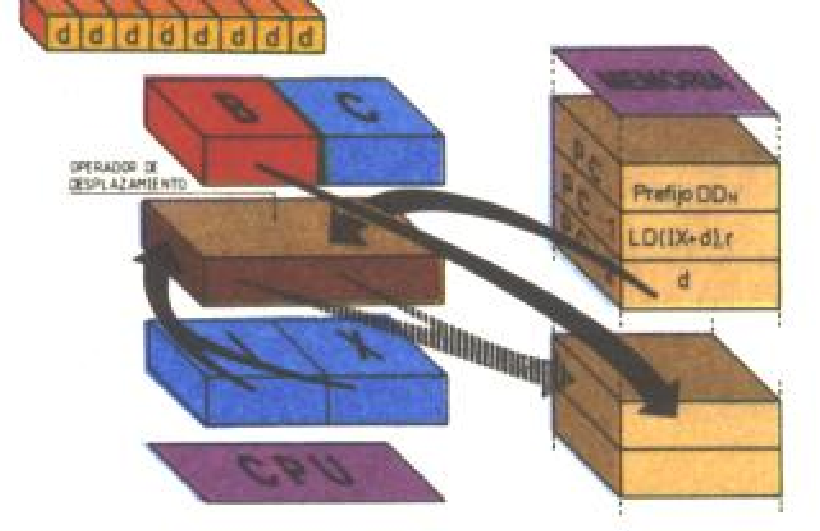
1 1 0 1 1 1 0 1

0 1 1 1 0 r r r

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



LD r,(IX+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d (d es un número de 8 bits en complemento a 2), es transferido a cualquier registro r.

Mnemónico: LD **Operandos:** r,(IX+d)

Formato binario:

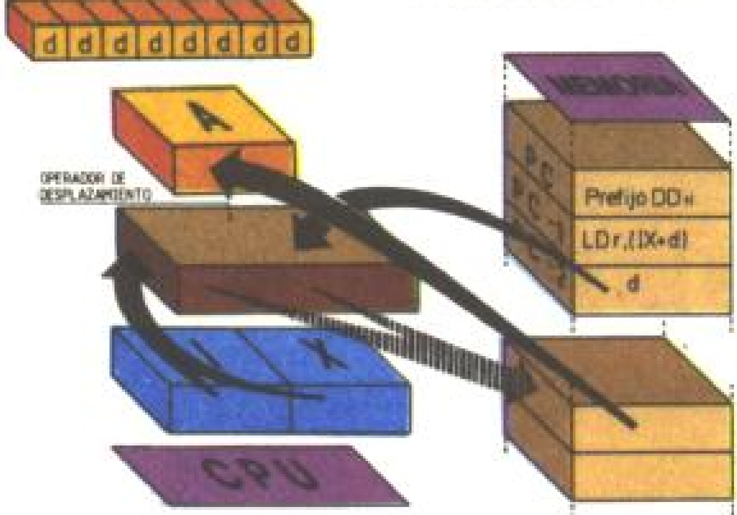
1 1 0 1 1 1 0 1

0 1 r r r 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



LD (IY+d),n   LD (IY+d),r   LD r,(IY+d)

LD (IY+d),n

El número de 8 bits n es transferido a la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d (d es un número de 8 bits en complemento a 2).

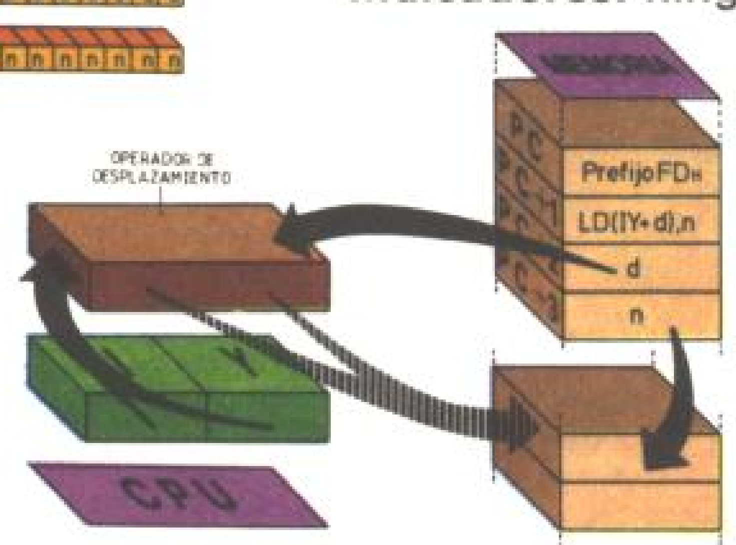
Mnemónico: LD   **Operandos:** (IY+d),n

Formato binario:

1	1	1	1	1	1	0	1
0	0	1	1	0	1	1	0
d	d	d	d	d	d	d	d
n	n	n	n	n	n	n	n

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



Instr.	Hex.	Dec.
LD (IY+d),n	FD,36,d,n	253,54,d,n
LD (IY+d),A	FD,77,d	253,119,d
LD (IY+d),B	FD,70,d	253,112,d
LD (IY+d),C	FD,71,d	253,113,d
LD (IY+d),D	FD,72,d	253,114,d
LD (IY+d),E	FD,73,d	253,115,d
LD (IY+d),H	FD,74,d	253,116,d
LD (IY+d),L	FD,75,d	253,117,d
LD A,(IY+d)	FD,7E,d	253,126,d
LD B,(IY+d)	FD,46,d	253,70,d
LD C,(IY+d)	FD,4E,d	253,78,d
LD D,(IY+d)	FD,56,d	253,86,d
LD E,(IY+d)	FD,5E,d	253,94,d
LD H,(IY+d)	FD,66,d	253,102,d
LD L,(IY+d)	FD,6E,d	253,110,d

LD (IY+d),r

El contenido de cualquier registro r es transferido a la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d (d es un número de 8 bits en complemento a 2).

Mnemónico: LD **Operandos:** (IY+d),r

Formato binario:

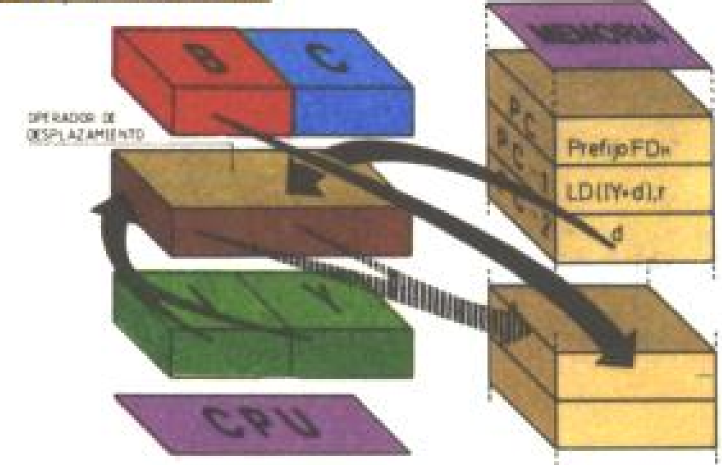


0 1 1 1 0 r r r



Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



LD r,(IY+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d (d es un número de 8 bits en complemento a 2), es transferido a cualquier registro r.

Mnemónico: LD **Operandos:** r,(IY+d)

Formato binario:

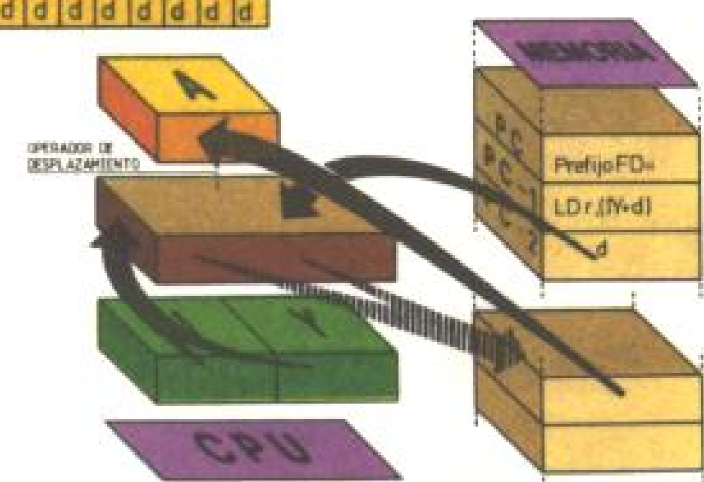
1 1 1 1 1 1 0 1

0 1 r r r 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ninguno



LD dd,nn LD IX,nn LD IY,nn

LD dd,nn

El número nn de 2 bytes, es transferido al par de registros especificado por el operando dd.

Nnemónico: LD **Operandos:** dd,nn

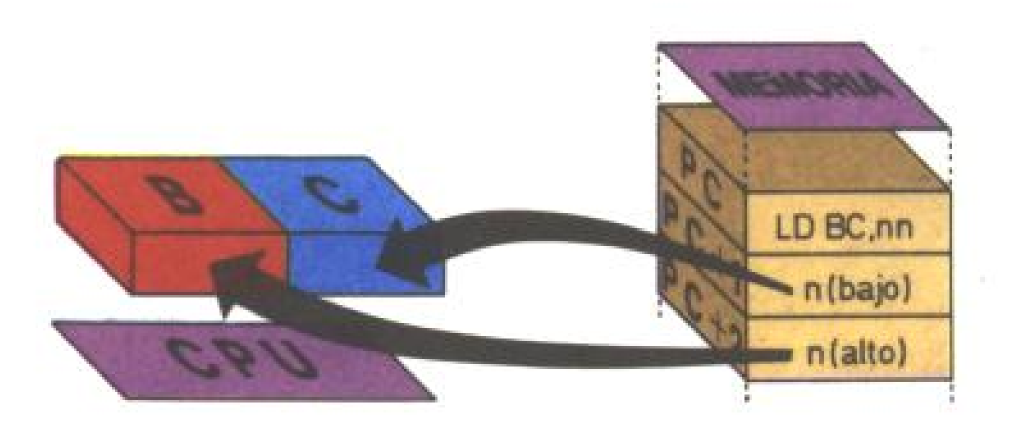
Formato binario:

0 0 d d 0 0 0 1
n n n n n n n n
n n n n n n n n

Ciclos: 3
Estados: 10 (4,3,3)

Indicadores: ninguno

Instr.	Hex.	Dec.
LD BC,nn	01,n,n	1,n,n
LD DE,nn	11,n,n	17,n,n
LD HL,nn	21,n,n	33,n,n
LD SP,nn	31,n,n	49,n,n
LD IX,nn	DD,21,n,n	221,33,n,n
LD IY,nn	FD,21,n,n	253,33,n,n



Ejemplo:

Después de ejecutar la instrucción
LD BC,4000H
resultará que el par BC contiene 4000H.

El código del par dd, para la construcción del código binario de la instrucción es:

BC	00
DE	01
HL	10
SP	11

LD IX,nn

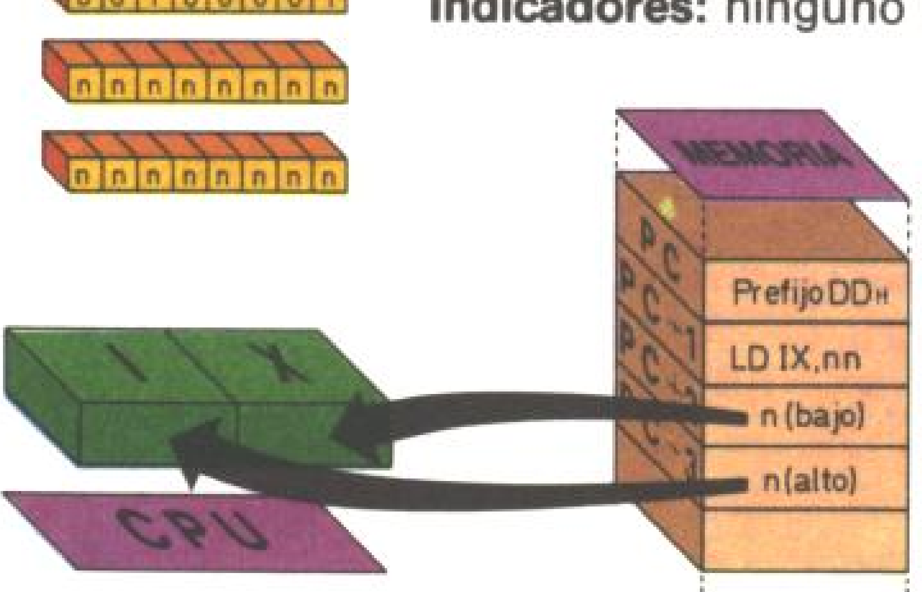
El número nn de 2 bytes, es transferido al par IX.

Mnemónico: LD **Operandos:** IX,nn

Formato binario:

Ciclos: 4
Estados: 14 (4,4,3,3)

Indicadores: ninguno



LD IY,nn

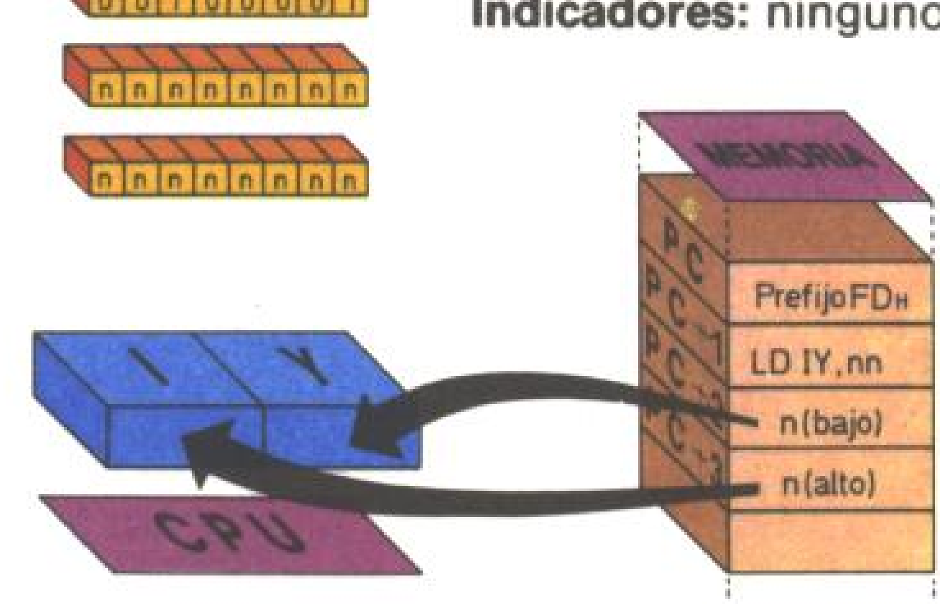
El número nn de 2 bytes, es transferido al par IY.

Mnemónico: LD **Operandos:** IY,nn

Formato binario:

Ciclos: 4
Estados: 14 (4,4,3,3)

Indicadores: ninguno



LD HL,(nn)

El contenido de la dirección de memoria especificada por el número nn de 2 bytes, es transferido al registro L, y el contenido de la siguiente dirección de memoria transferido al registro H.

Instr.	Hex.	Dec.
LD HL,(nn)	2A,n,n	42,n,n
LD BC,(nn)	ED,4B,n,n	237,75,n,n
LD DE,(nn)	ED,5B,n,n	237,91,n,n
LD HL,(nn)	ED,6B,n,n	237,107,n,n
LD SP,(nn)	ED,7B,n,n	237,123,n,n
LD IX,(nn)	DD,2A,n,n	221,42,n,n
LD IY,(nn)	FD,2A,n,n	253,42,n,n

Mnemónico: LD **Operandos:** HL,(nn)

Formato binario:

0 0 1 0 1 0 1 0

n n n n n n n n

n n n n n n n n

Ciclos: 5
Estados: 16 (4,3,3,3,3,)

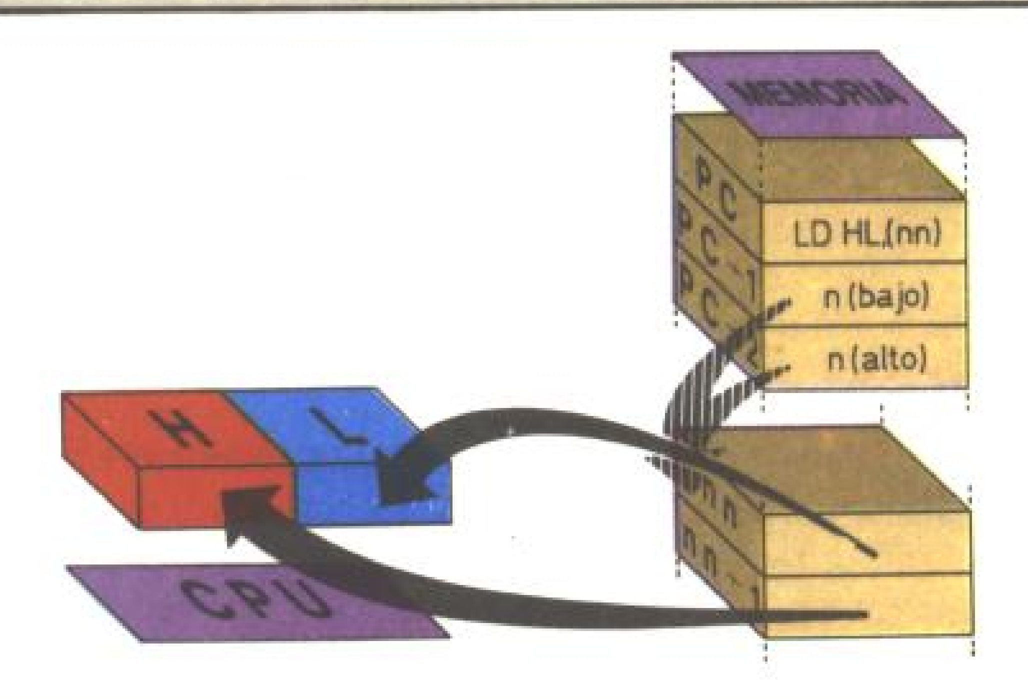
Indicadores: ninguno

Ejemplo:

Si el contenido de la dirección de memoria 7FF4H es 00H y el contenido de la dirección de memoria 7FF5H es FFH, después de ejecutar la instrucción

LD HL,(7FF4H)

resultará que el par HL contiene FF00H.



LD dd,(nn)

El contenido de la dirección de memoria especificada por el número nn de 2 bytes, es transferido al registro bajo del par especificado por el operando dd, que puede ser BC, DE, HL o SP, y el contenido de la siguiente dirección de memoria es transferido al registro alto de dicho par.

Mnemónico: LD
Formato binario:

11101101
01dd1011
nnnnnnnn
nnnnnnnn

Operandos: dd,(nn)

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninguno

LD IX, (nn)

El contenido de la dirección de memoria especificada por el número nn de 2 bytes, es transferido al registro bajo del par IX, y el contenido de la siguiente dirección de memoria es transferido al registro alto de dicho par.

Mnemónico: LD
Formato binario:

11011101
00101010
nnnnnnnn
nnnnnnnn

Operandos: IX,(nn)

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninguno

LD IY,(nn)

El contenido de la dirección de memoria especificada por el número nn de 2 bytes, es transferido al registro bajo del par IY, y el contenido de la siguiente dirección de memoria es transferido al registro alto de dicho par.

Mnemónico: LD
Formato binario:

11111101
00101010
nnnnnnnn
nnnnnnnn

Operandos: IX,(nn)

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninguno

LD (nn),HL

El contenido del registro L es transferido a la dirección de memoria especificada por el número nn de 2 bytes, y el contenido del registro H es transferido a la siguiente dirección de memoria.

Nnemónico: LD **Operandos:** (nn),HL

Formato binario:

0 0 1 0 0 0 1 0

n n n n n n n n



Ciclos: 5
Estados: 16(4,3,3,3,3)

Indicadores: ninguno

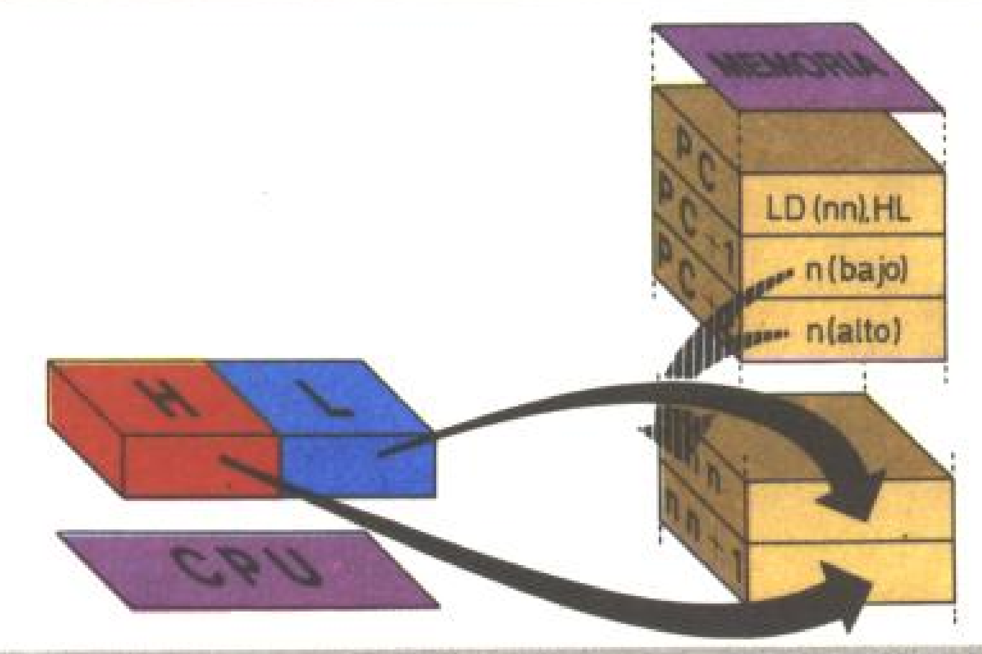
Ejemplo:

Si el contenido del par HL es 1234H, después de ejecutar la instrucción

LD (FF00H), HL

resultará que la dirección de memoria FF00H contiene 34H, y la dirección de memoria FF01H contiene 12H.

Instr.	Hex.	Dec.
LD (nn),HL	22,n,n	34,n,n
LD (nn),BC	ED,43,n,n	237,67,n,n
LD (nn),DE	ED,53,n,n	237,83,n,n
LD (n,n),HL	ED,63,n,n	237,99,n,n
LD (nn),SP	ED,73,n,n	237,115,n,n
LD (nn),IX	DD,22,n,n	221,34,n,n
LD (nn),IY	FD,22,n,n	253,34,n,n



LD (nn),dd

El contenido del registro del par especificado por el operando dd, que puede ser BC, DE, HL o SP, es transferido a la dirección de memoria especificada por el número nn de 2 bytes, y el contenido del registro alto de dicho para es transferido a la siguiente dirección de memoria.

Mnemónico: LD
Formato binario:

1 1 1 0 1 1 0 1
0 1 d d 0 0 1 1
n n n n n n n n
n n n n n n n n

Operandos: (nn),dd

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninquno

LD (nn), IX

El contenido del registro bajo del par IX es transferido a la dirección especificada por el número nn de 2 bytes, y el contenido del registro alto de dicho par es transferido a la siguiente dirección de memoria.

Mnemónico: LD
Formato binario:

1 1 0 1 1 1 0 1
0 0 1 0 0 0 1 0
n n n n n n n n
n n n n n n n n

Operandos: (nn),IX

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninguno

LD (nn), IY

El contenido del registro bajo del par IY es transferido a la dirección de memoria especificada por el número nn de 2 bytes, y el contenido del registro alto de dicho par es transferido a la siguiente dirección de memoria.

Mnemónico: LD
Formato binario:

1 1 1 1 1 1 0 1
0 0 1 0 0 0 1 0
n n n n n n n n
n n n n n n n n

Operandos: (nn),IY

Ciclos: 6
Estados: 20 (4,4,3,3,3,3)

Indicadores: ninguno

LD SP,HL   LD SP,IX   LD SP,IY

LD SP,HL

El contenido del par HL es transferido al par SP.

Mnemónico: LD   **Operandos:** SP,HL

Formato binario:

1 1 1 1 1 0 0 1

Ciclos: 1
Estados: 6

Indicadores: ninguno

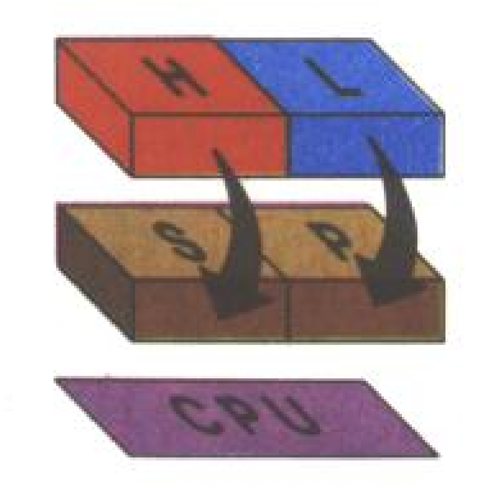
Instr.	**Hex.**	**Dec.**
LD SP,HL	F9	249
LD SP,IX	DD,F9	221,249
LD SP,IY	FD,F9	253,249

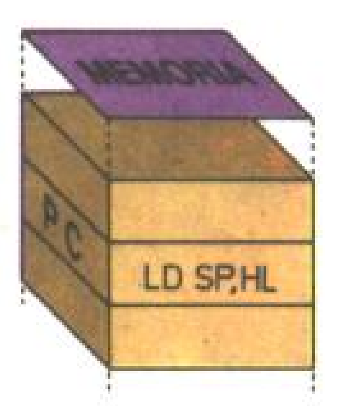
Ejemplo:

Si el contenido del par HL es 9000H, después de ejecutar la instrucción

LD SP,HL

resultará que el par SP contiene 9000H.





LD SP,IX

El contenido del par IX es transferido al par SP.

Mnemónico: LD **Operandos:** SP,IX

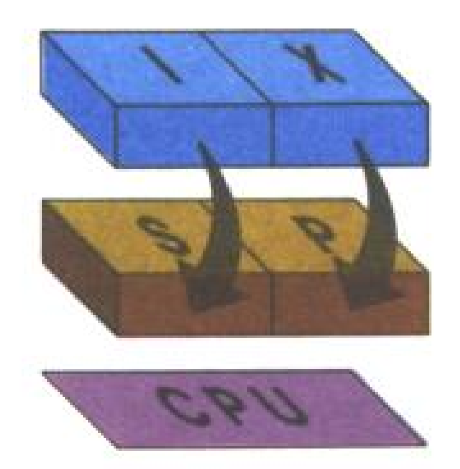
Formato binario:

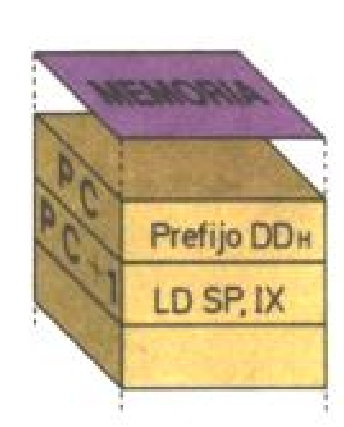
1 1 0 1 1 1 0 1

1 1 1 0 0 0 1 1

Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





LD SP, IY

El contenido del par IY es transferido al par SP.

Mnemónico: LD **Operandos:** SP,IY

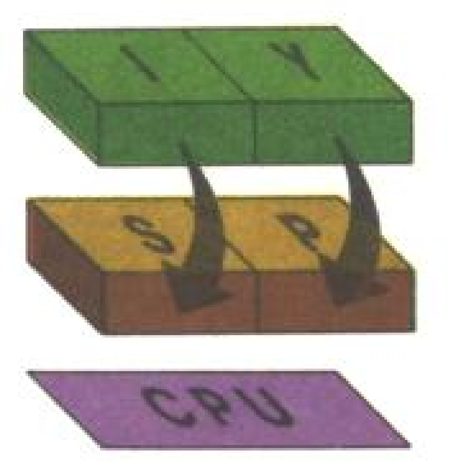
Formato binario:

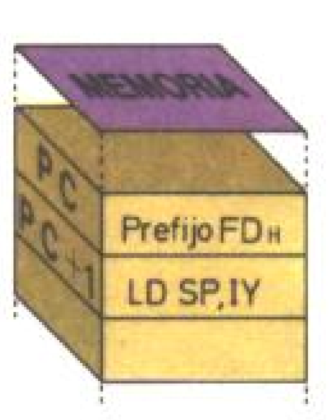
1 1 1 1 1 1 0 1

1 1 1 1 1 0 0 1

Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





EXX

El contenido de los pares BC, DE y HL es intercambiado con el contenido de los mismos pares del grupo alternativo de registros.

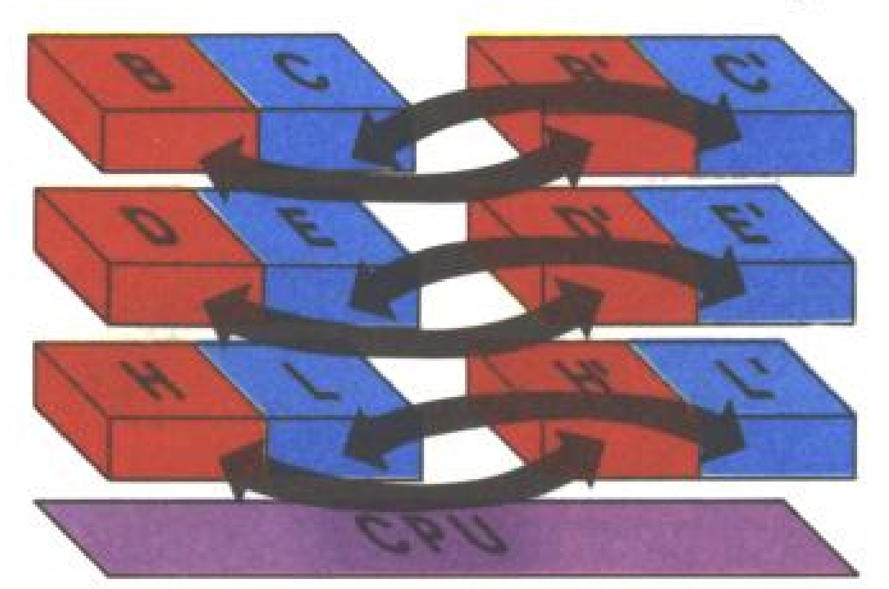
Mnemónico: EXX **Operandos:** no tiene

Formato binario:



Ciclos: 1
Estados: 4

Indicadores: ninguno



Instr.	Hex.	Dec.
EXX	D9	217
EX DE,HL	EB	235
EX AF,AF'	08	8

Ejemplo:

Si el contenido de los pares de registros está de la siguiente manera:

BC: 0000H	BC': 3333H
DE: 1111H	DE': 4444H
HL: 2222H	HL': 5555H

después de ejecutar la instrucción

EXX

resultará que los pares contienen:

BC: 3333H	BC': 0000H
DE: 4444H	DE': 1111H
HL: 5555H	HL': 2222H

EX DE,HL

El contenido de los pares DE y HL es intercambiado.

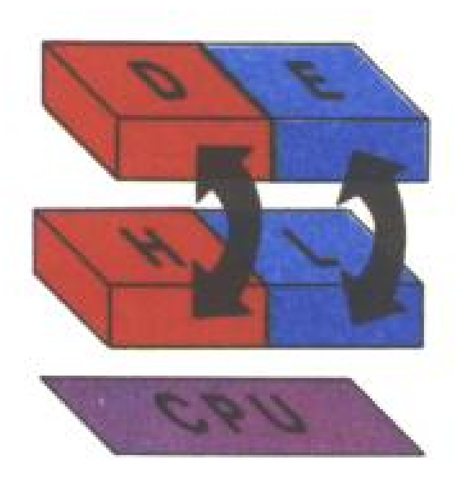
Mnemónico: EX **Operandos:** DE,HL

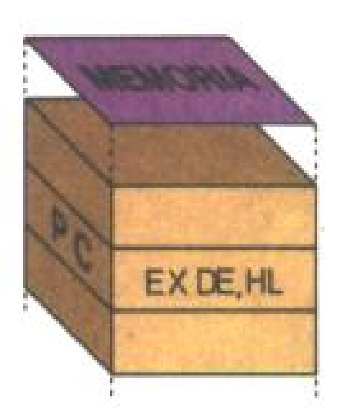
Formato binario:

1	1	1	0	1	0	1	1

Ciclos: 1
Estados: 4

Indicadores: ninguno





EX AF, AF'

El contenido del par AF es intercambiado con el contenido del mismo par del grupo alternativo de registros.

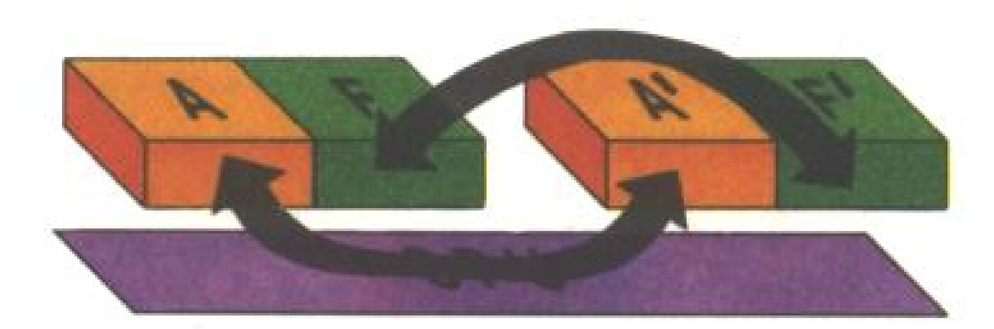
Mnemónico: EX **Operandos:** AF,AF'

Formato binario:

0	0	0	0	1	0	0	0

Ciclos: 1
Estados: 4

Indicadores: ninguno



EX (SP),HL

El contenido de la dirección de memoria apuntada por el par SP es intercambiado por el contenido del registro L, y el contenido de la siguiente dirección de memoria es intercambiado con el contenido del registro H.

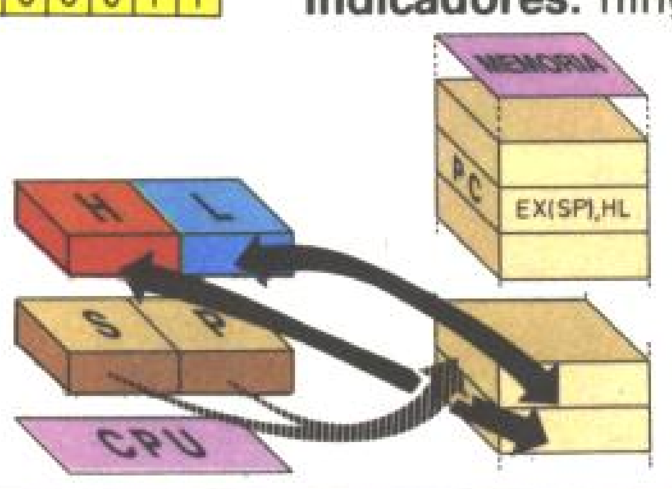
Mnemónico: EX **Operandos:** (SP),HL

Formato binario:

Ciclos: 5
Estados: 19 (4,3,4,3,5)



Indicadores: ninguno



Instr.	Hex.	Dec.
EX (SP),HL	E3	227
EX (SP),IX	DD,E3	221,227
EX (SP),IY	FD,E3	253,227

Ejemplo:

Si el contenido del par HL es 0100H, el contenido del par SP es 70A0H, el contenido de la dirección de memoria 70A0H es 50H, y el contenido de la dirección de memoria 70A1H es 05H, después de ejecutar la instrucción

EX (SP),HL

resultará que el par HL contiene 0550H, la dirección de memoria 70A0H contiene 00H, la dirección de memoria 70A1H contiene 01H, y el par SP no cambia.

EX (SP),IX

El contenido de la dirección de memoria apuntada por el par SP es intercambiado con el contenido bajo del par IX, y el contenido de la siguiente dirección de memoria es intercambiado con el contenido alto del par IX.

Mnemónico: EX **Operandos:** (SP),IX

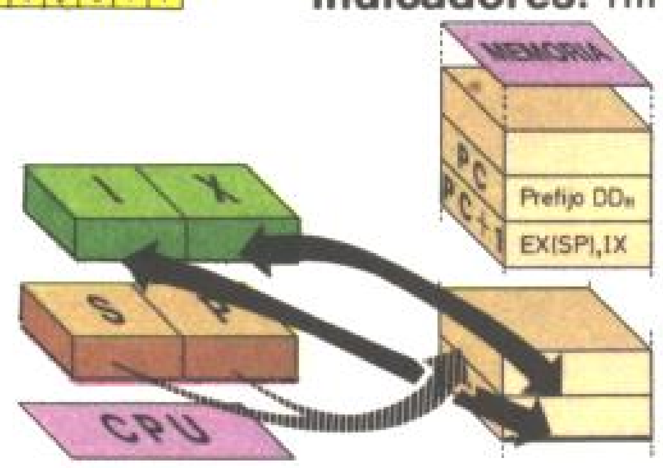
Formato binario:

11011101

11100011

Ciclos: 6
Estados: 23 (4,4,3,3,3,5)

Indicadores: ninguno



EX (SP),IY

El contenido de la dirección de memoria apuntada por el par SP es intercambiado con el contenido bajo del par IY, y el contenido de la siguiente dirección de memoria es intercambiado con el contenido alto del par IY.

Mnemónico: EX **Operandos:** (SP),IY

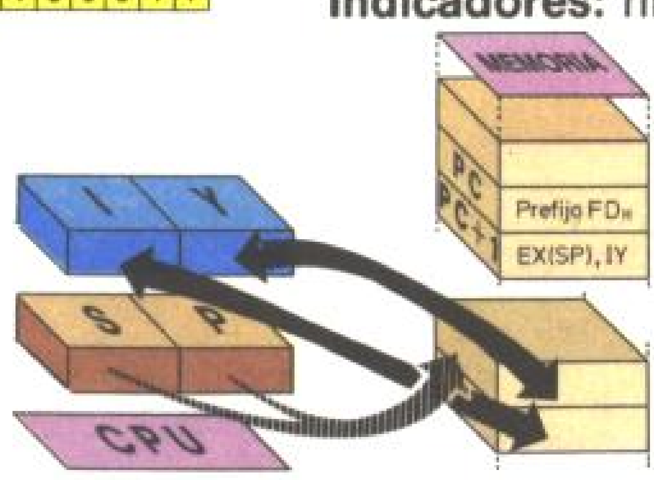
Formato binario:

11111101

11100011

Ciclos: 4
Estados: 23 (4,4,3,4,3,5)

Indicadores: ninguno



ADD A,r

El contenido de cualquier registro r es sumado con el contenido del registro A, en el cual queda el resultado

Mnemónico: ADD   **Operandos:** A,r

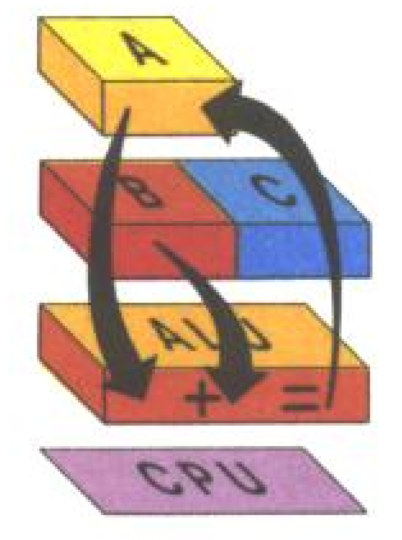
Formato binario:

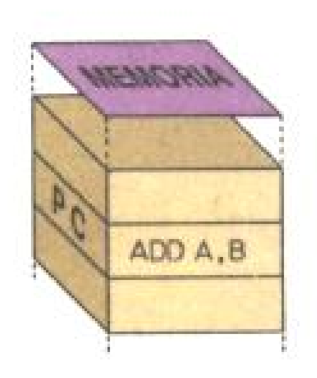
1	0	0	0	0	r	r	r

Ciclos: 1
Estados: 4

Indicadores: ver tabla

Instr.	Hex.	Dec.
ADD A,A	87	135
ADD A,B	80	128
ADD A,C	81	129
ADD A,D	82	130
ADD A,E	83	131
ADD A,H	84	132
ADD A,L	85	133
ADD A,n	C6,n	198,n





Ejemplo:

Si el registro B contiene 7AH, y el registro A contiene 12H, después de ejecutar la instrucción

ADD A,B

resultará que el registro A contiene 8CH (7AH + 12H), y el registro B conserva el anterior valor de 7AH.

ADD A,n

El número n de 8 bits es sumado al contenido del registro A, en el cual queda el resultado.

Mnemónico: ADD **Operandos:** A,n

Formato binario:

1 1 0 0 0 1 1 0

n n n n n n n n

Ciclos: 2
Estados: 7 (4,3)
Indicadores: ver tabla

Tabla de indicadores:

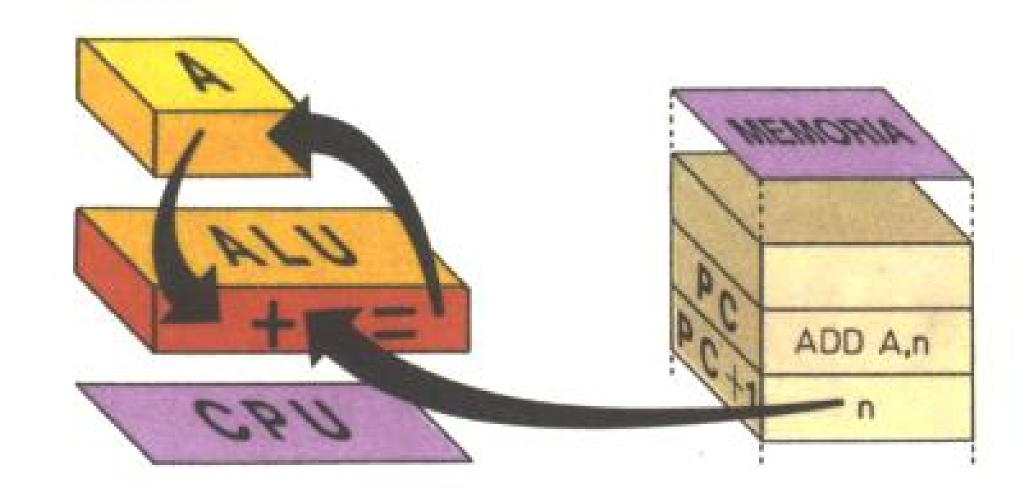
S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 0
C a 1 si hay acarreo del bit 7

Ejemplo:

Si el registro A contiene 50 H, después de ejecutar la instrucción

ADD A,15H

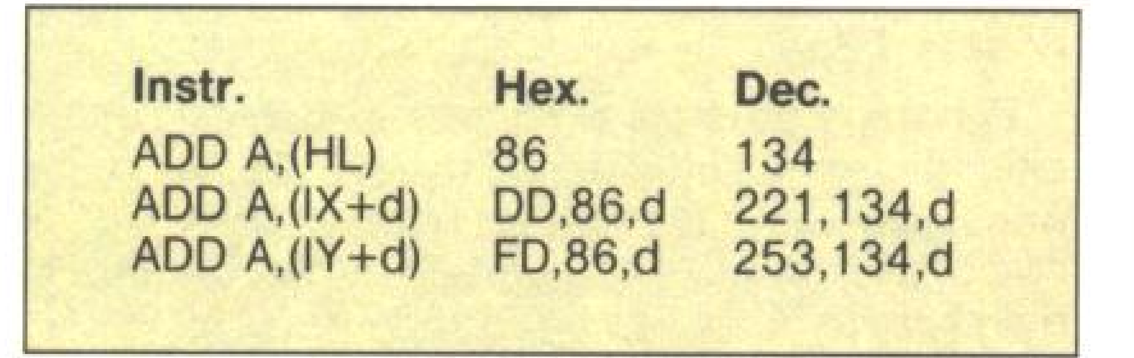
resultará que el registro A contiene 65H (50H + 15H).



ADD A, (HL) ADD A, (IX+d) ADD A,(IY+d)

ADD A,(HL)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del par HL es sumado con el contenido del registro A, en el cual queda el resultado.



Instr.	Hex.	Dec.
ADD A,(HL)	86	134
ADD A,(IX+d)	DD,86,d	221,134,d
ADD A,(IY+d)	FD,86,d	253,134,d

Mnemónico: ADD

Operandos: A,(HL)

Formato binario:

1	0	0	0	0	1	1	0

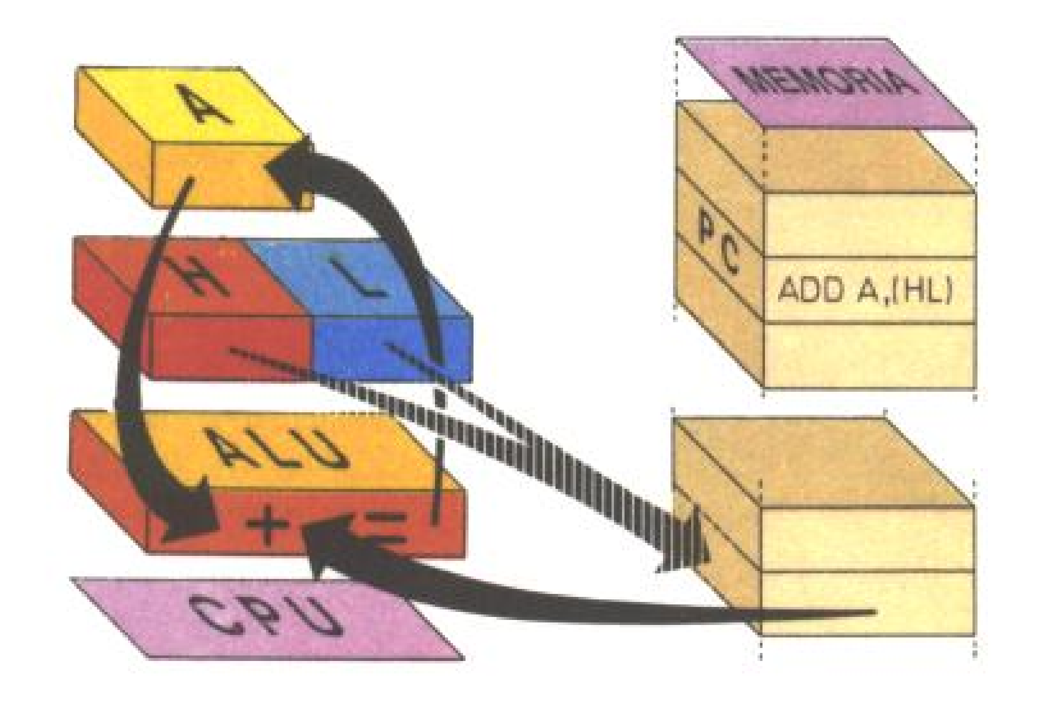
Ciclos: 2

Estados: 7 (4,3)

Indicadores: ver tabla

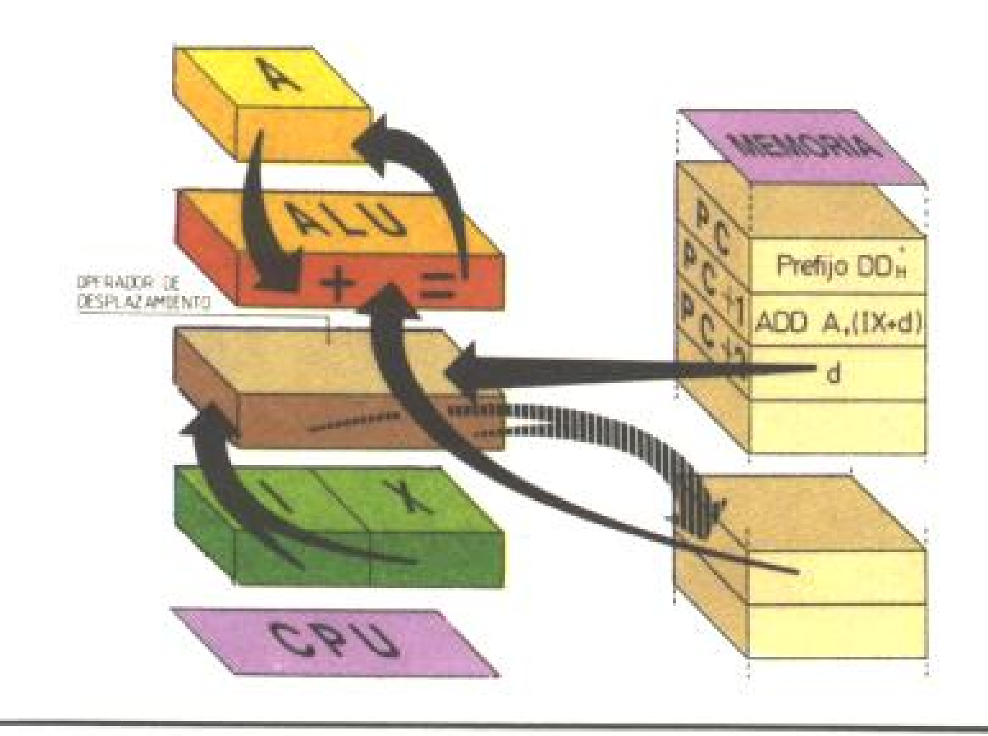
Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1 si hay acarreo del bit 3
P/V	a 1 si hay exceso
N	a 0
C	a 1 si hay acarreo del bit 7



ADD A, (IX+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d (d es un número de 8 bits en complemento a 2), es sumado con el contenido del registro A, en el cual queda el resultado.



Mnemónico: ADD

Operandos: A,(IX+d)

Formato binario:

1 1 0 1 1 1 0 1

1 0 0 0 0 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

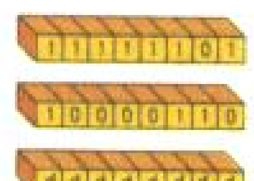
ADD A,(IY+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d (d es un número de 8 bits en complemento a 2), es sumado con el contenido del registro A, en el cual queda el resultado.

Mnemónico: ADD

Operandos: A,(IY+d)

Formato binario:



Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

ADC A,r

El contenido de cualquier registro r es sumado con el indicador de acarreo y con el contenido del registro A, en el cual queda el resultado.

Mnemónico: ADC

Operandos: A,r

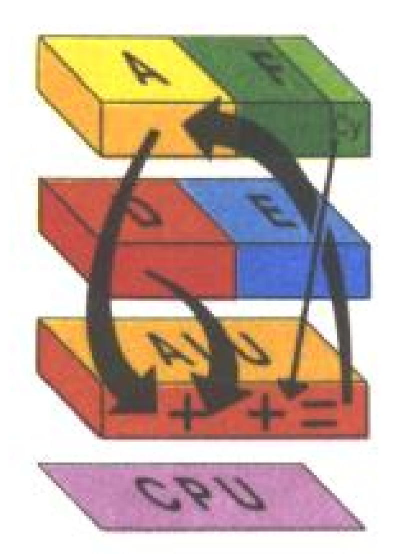
Formato binario:

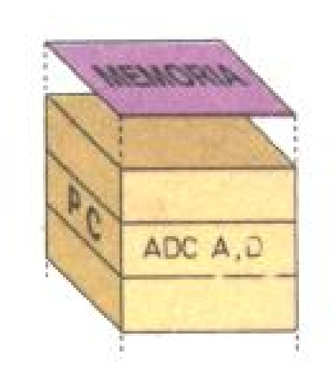


Ciclos: 1
Estados: 4

Indicadores: ver tabla

Instr.	Hex.	Dec.
ADC A,A	8F	143
ADC A,B	88	136
ADC A,C	89	137
ADC A,D	8A	138
ADC A,E	8B	139
ADC A,H	8C	140
ADC A,L	8D	141
ADC A,n	CE,n	206,n





Ejemplo:

Si el registro D contiene 2 FH, el registro A tiene 00H, y el indicador de acarreo está activado (CY=1), después de ejecutar la instrucción

ADC A,D

resultará que el registro A contiene 30H (2FH + 00H + 1H), y el indicador de acarreo quedará desactivado (CY=0).

ADC A,n

El número n de 8 bits es sumado con el indicador de acarreo al contenido del registro A, en el cual queda el resultado.

Mnemónico: ADC **Operandos:** A,n

Formato binario:

1 1 0 0 1 1 1 0

n n n n n n n n

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

Ejemplo:

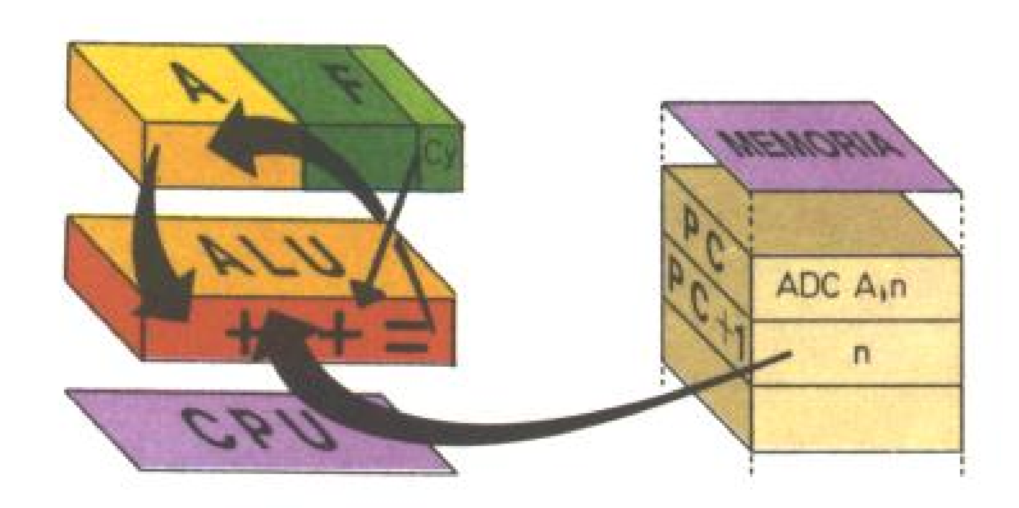
Si el registro A contiene 01H, y el indicador de acarreo está desactivado (CY=0), después de ejecutar la instrucción

ADC A,FFH

resultará que el registro A contiene 00H, y el indicador de acarreo quedará a su vez activado (CY=1), porque 01H + FFH + 00H = 100H.

Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1 si hay acarreo del bit 3
P/V	a 1 si hay exceso
N	a 0
C	a 1 si hay acarreo del bit 7



ADC A,(HL) ADC A,(IX+d) ADC A,(IY+d)

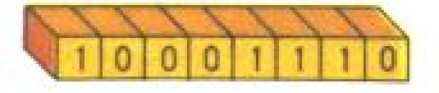
ADC A, (HL)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del par HL es sumado con el indicador de acarreo y con el contenido del registro A, en el cual queda el resultado.

Instr.	Hex.	Dec.
ADC A,(HL)	8E	142
ADC A,(IX+d)	DD,8E,d	221,142,d
ADC A,(IY+d)	FD,8E,d	253,142,d

Mnemónico: ADC **Operandos:** A,(HL)

Formato binario:

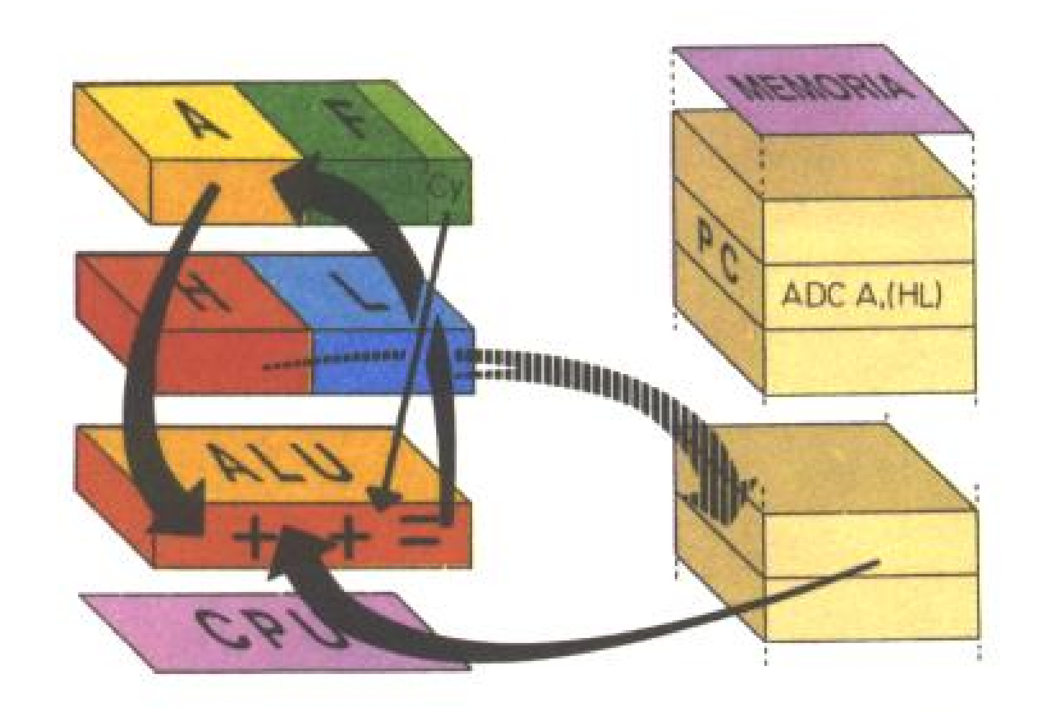


Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

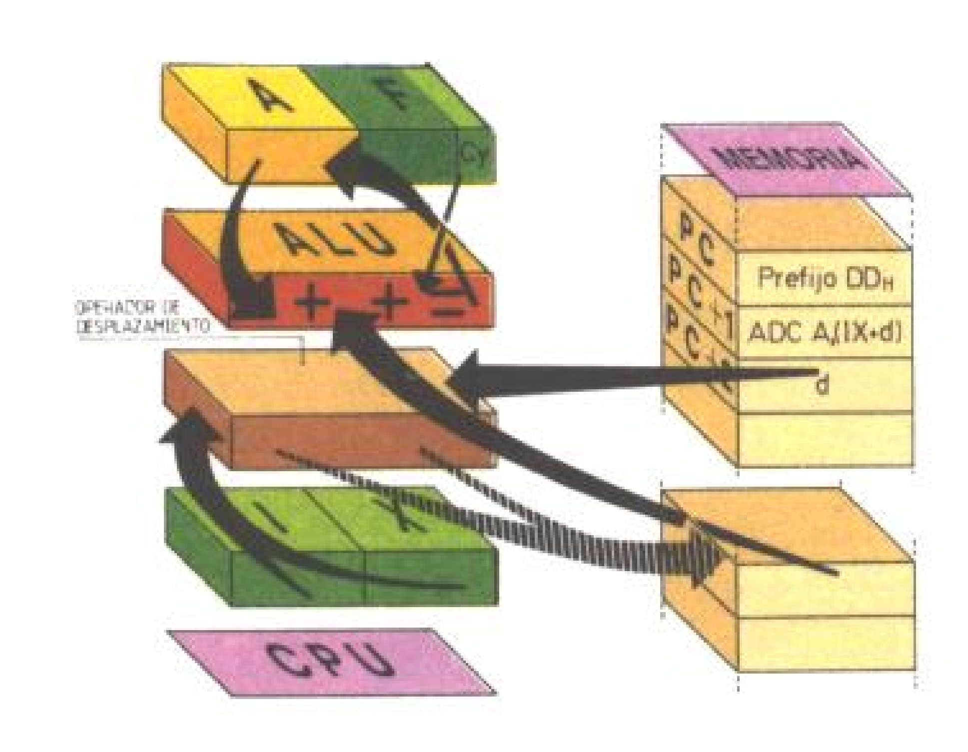
Tabla de indicadores:

S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 0
C a 1 si hay acarreo del bit 7



ADC A,(IX+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d es sumado con el indicador de acarreo y con el contenido del registro A, en el cual queda el resultado.



Mnemónico: ADC

Operandos: A,(IX+d)

Formato binario:

1	1	0	1	1	1	0	1
1	0	0	0	1	1	1	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

ADC A,(IY+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d es sumado con el indicador de acarreo y con el contenido del registro A, en el cual queda el resultado.

Mnemónico: ADC

Operandos: A,(IY+d)

Formato binario:

1	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

SUB r SUB n

SUB r

El contenido de cualquier registro r es restado al contenido del registro A, en el cual queda el resultado.

Mnemónico: SUB **Operando:** r

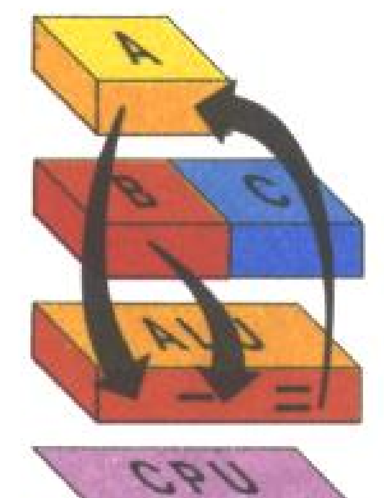
Formato binario:

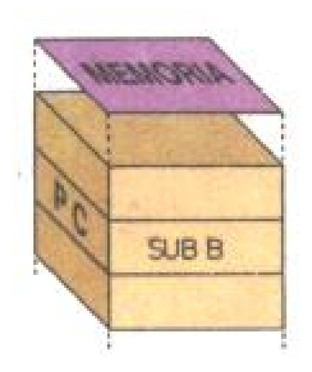
1 0 0 1 1 r r r

Ciclos: 1
Estados: 4

Indicadores: ver tabla

Instr.	Hex.	Dec.
SUB A	97	151
SUB B	90	144
SUB C	91	145
SUB D	92	146
SUB E	93	147
SUB H	94	148
SUB L	95	149
SUB n	D6,n	214,n





Ejemplo:

Si el registro B contiene 12 H, y el registro A contiene 7AH, después de ejecutar la instrucción

SUB B

resultará que el registro A contiene 68H, y el registro B conserva el anterior valor de 12H.
(7AH – 12H = 68H)

SUB n

El número n de 8 bits es restado al contenido del registro A, en el cual queda el resultado.

Mnemónico: SUB **Operando:** n

Formato binario:

1	1	0	1	1	1	1	0
n	n	n	n	n	n	n	n

Ciclos: 2
Estados: 7 (4,3)
Indicadores: ver tabla

Tabla de indicadores:

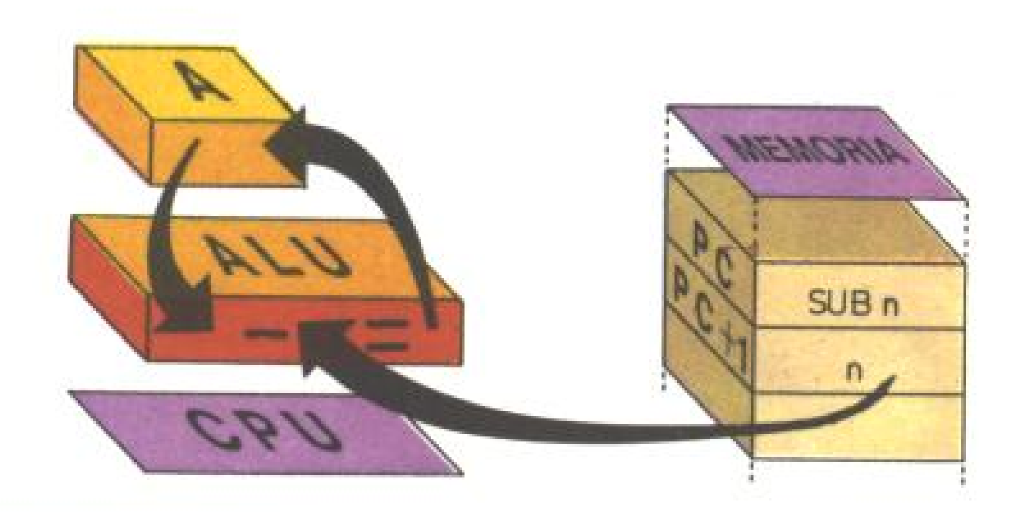
S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 1
C a 1 si hay acarreo del bit 7

Ejemplo:

Si el registro A contiene 50H, después de ejecutar la instrucción

SUB 11H

resultará que el registro A contiene 3FH.
(50H − 11H = 3FH)



SUB (HL) SUB (IX+d) SUB (IY+d)

SUB (HL)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del par HL es restado al contenido del registro A, en el cual queda el resultado.

Instr.	Hex.	Dec.
SUB (HL)	96	150
SUB (IX+d)	DD,96,d	221,150,d
SUB (IY+d)	FD,96,d	253,150,d

Mnemónico: SUB **Operando:** (HL)

Formato binario:

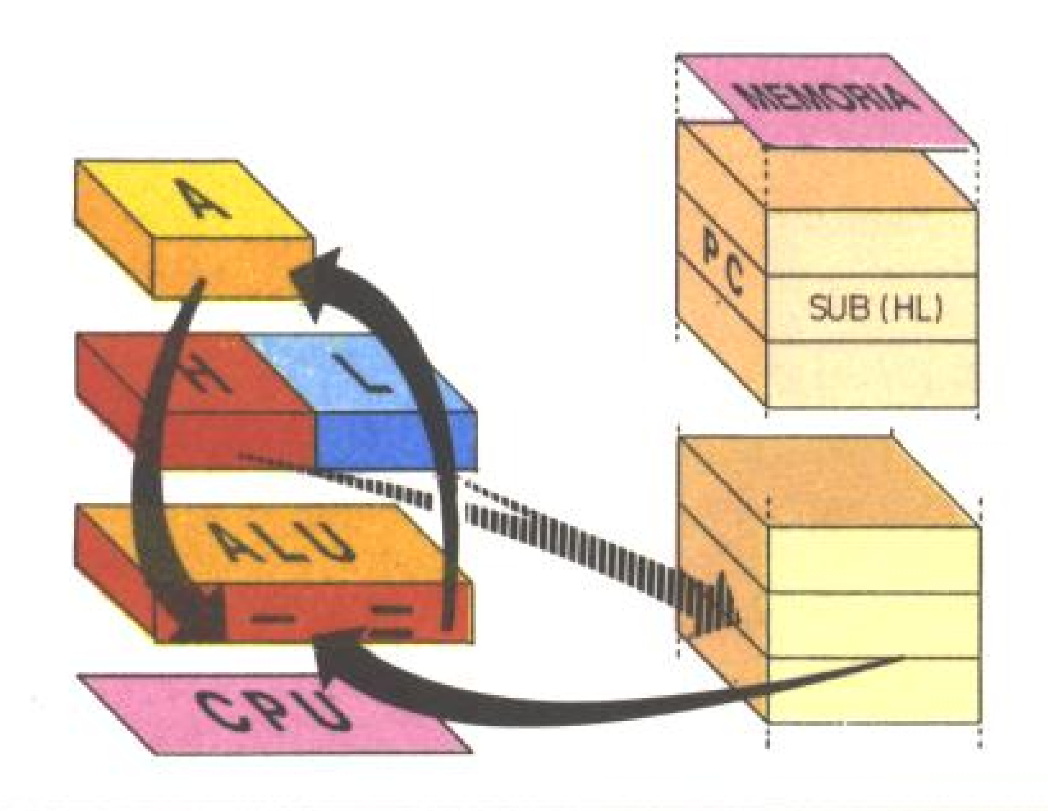
1	0	0	1	0	1	1	0

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

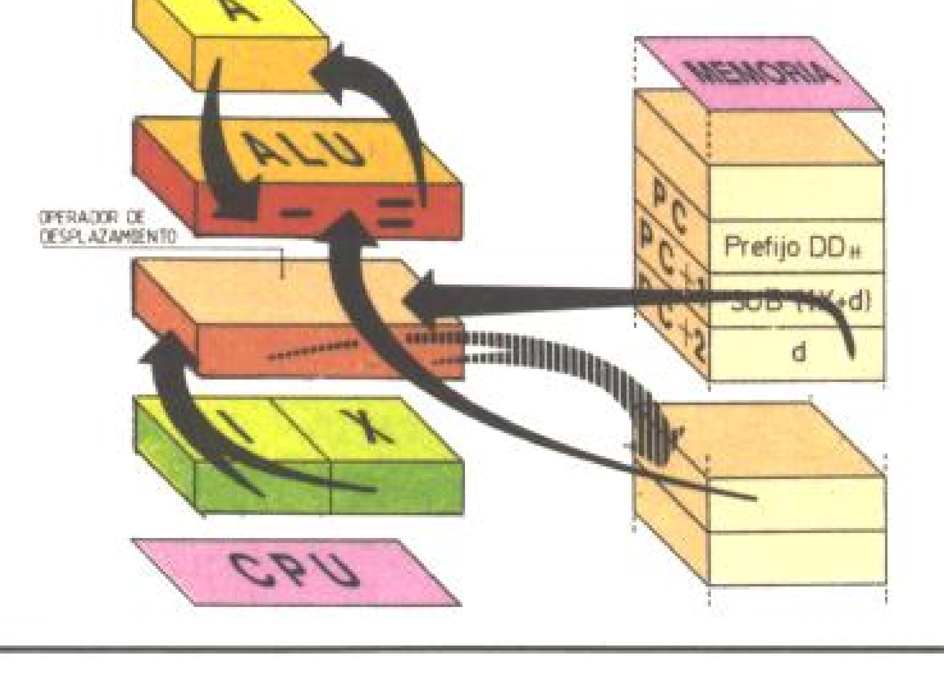
Tabla de indicadores:

S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 1
C a 1 si hay acarreo del bit 7



SUB (IX+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d, es restado al contenido del registro A, en el cual queda el resultado.



Mnemónico: SUB **Operando:** (IX+d)

Formato binario:

11011101
10010110
dddddddd

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

SUB (IY+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d es restado al contenido del registro A, en el cual queda el resultado.

Mnemónico: SUB **Operando:** (IY+d)

Formato binario:

11111101
10010110
dddddddd

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

SBC A,r

El contenido de cualquier registro r y el indicador de acarreo son restados al contenido del registro A, en el cual queda el resultado.

Mnemónico: SBC

Operandos: A,r

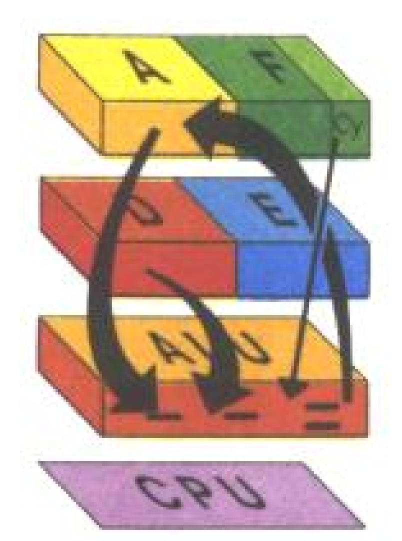
Formato binario:

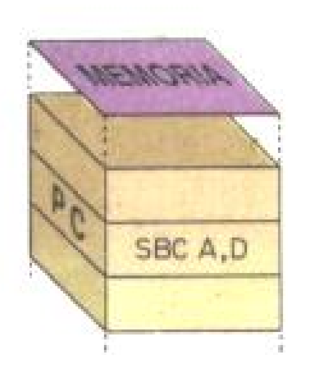
1 0 0 1 0 r r r

Ciclos: 1
Estados: 4

Indicadores: ver tabla

Instr.	Hex.	Dec.
SBC A,A	9F	159
SBC A,B	98	152
SBC A,C	99	153
SBC A,D	9A	154
SBC A,E	9B	155
SBC A,H	9C	156
SBC A,L	9D	157
SBC A,n	DE,n	222,n





Ejemplo:

Si el registro D contiene A6H, el registro A F8H, y el indicador de acarreo está desactivado (CY=0), después de ejecutar la instrucción

SBC A,D

resultará que el registro A contiene 52H, es decir: (F8H – A6H – 0H = 52H), y el indicador de acarreo quedará desactivado (Cy=0).

SBC A,n

El número n de 8 bits y el indicador de acarreo son restados al contenido del registro A, en el cual queda el resultado.

Mnemónico: SBC   **Operandos:** A,n

Formato binario:

1	1	0	1	0	1	1	0
n	n	n	n	n	n	n	n

Ciclos: 2
Estados: 7 (4 + 3)

Indicadores: ver tabla

Tabla de indicadores:

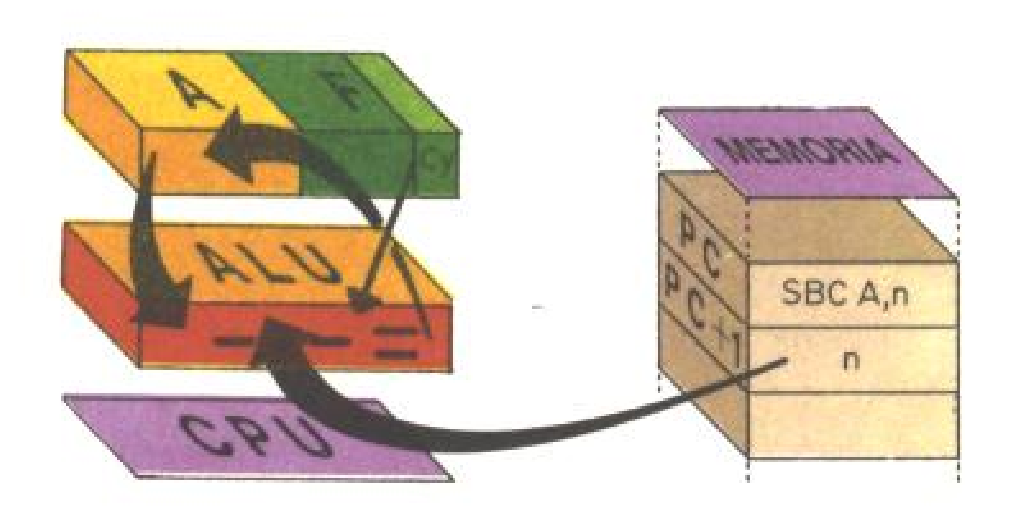
S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 1
C a 1 si hay acarreo del bit 7

Ejemplo:

Si el registro A contiene 01H, y el indicador de acarreo está activado (CY=1), después de ejecutar la instrucción

SBC A,15H

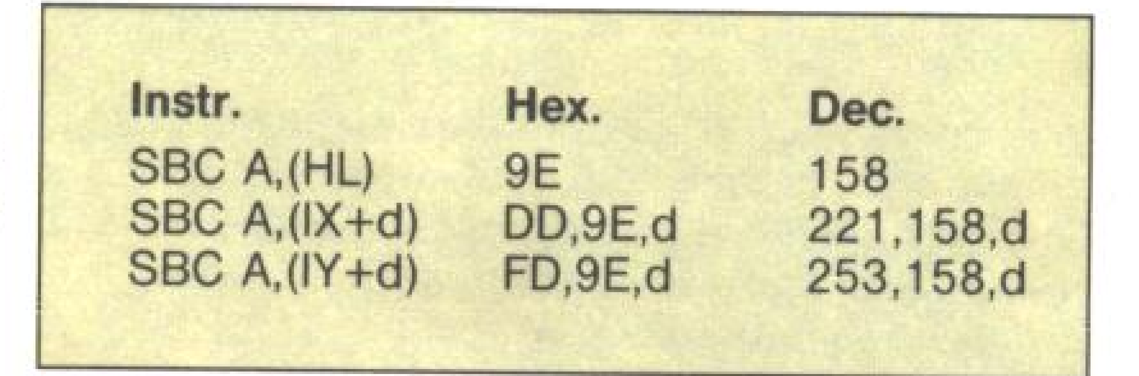
resultará que el registro A contiene FAH, y el indicador de acarreo quedará a su vez activado (CY=1), porque 10H – 15H = –06H y 100H – 06H = FAH.



SBC A,(HL) SBC A,(IX+d) SBC A,(IY+d)

SBC A,(HL)

El contenido de 8 bits de la dirección de memoria especificada por el contenido del par HL y el indicador de acarreo son restados al contenido del registro A, en el cual queda el resultado.



Instr.	Hex.	Dec.
SBC A,(HL)	9E	158
SBC A,(IX+d)	DD,9E,d	221,158,d
SBC A,(IY+d)	FD,9E,d	253,158,d

Mnemónico: SBC **Operandos:** A,(HL)

Formato binario:

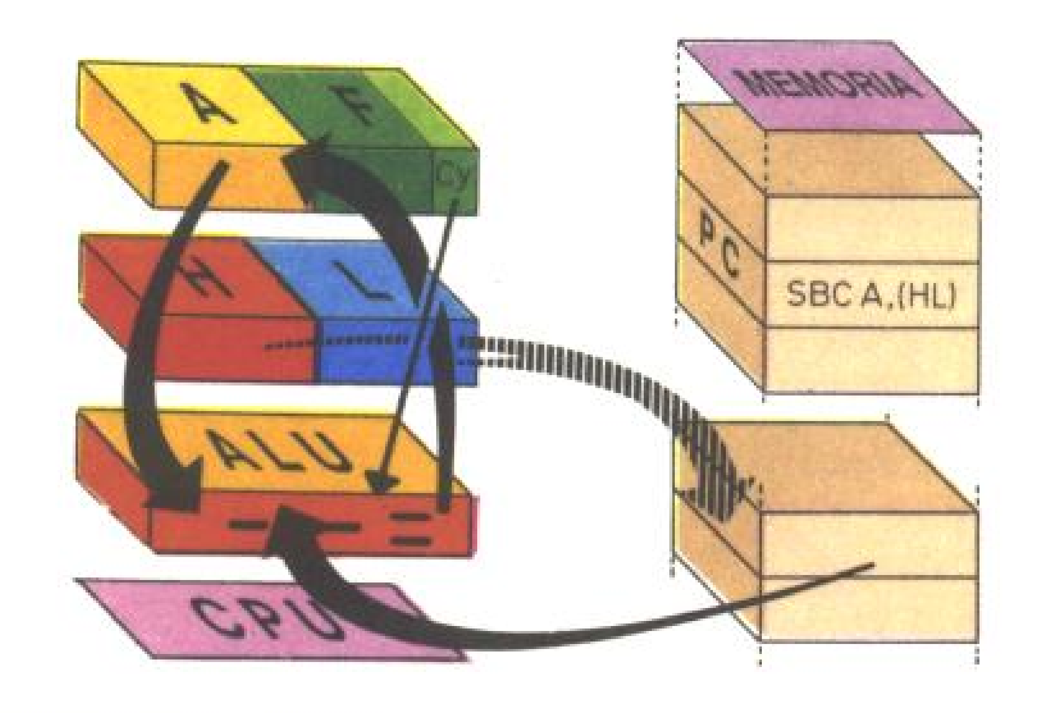
1 0 0 1 1 1 1 0

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

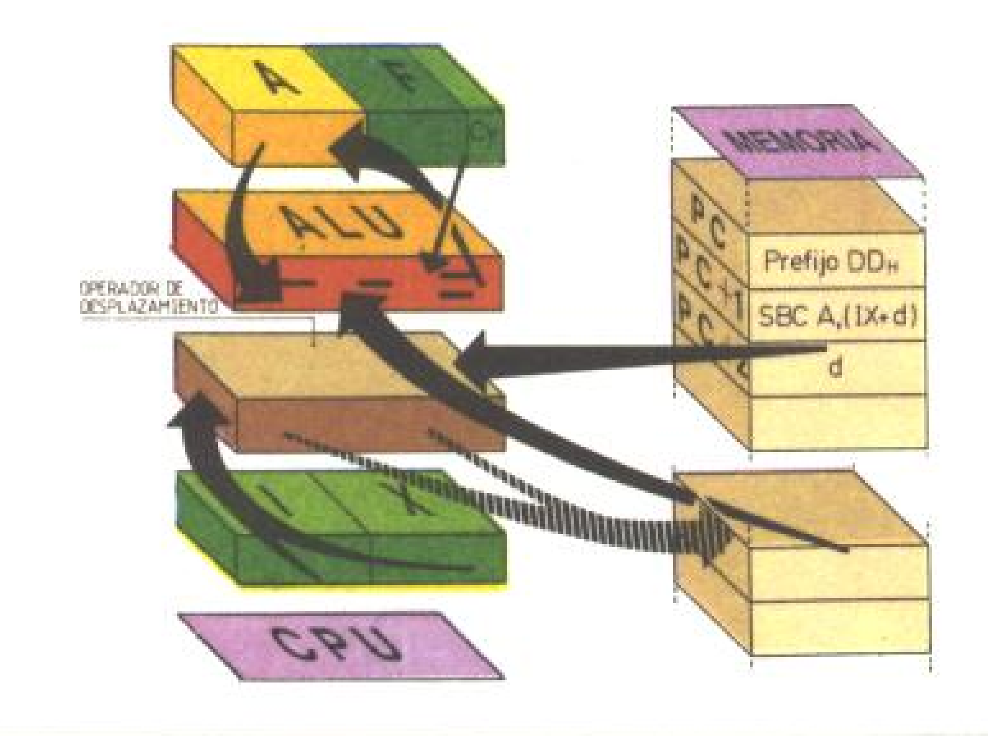
Tabla de indicadores:

S a 1 si el resultado es negativo
Z a 1 si el resultado es cero
H a 1 si hay acarreo del bit 3
P/V a 1 si hay exceso
N a 1
C a 1 si hay acarreo del bit 7



SBC A,(IX+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IX y el desplazamiento d y el indicador de acarreo son restados al contenido del registro A, en el cual queda el resultado.



Mnemónico: SBC

Operandos: A,(IX+d)

Formato binario:

1	1	0	1	1	1	0	1
1	0	0	1	1	1	1	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

SBC A,(IY+d)

El contenido de la dirección de memoria especificada por la suma del contenido del par IY y el desplazamiento d y el indicador de acarreo son restados al contenido del registro A, en el cual queda el resultado.

Mnemónico: SBC

Operandos: A,(IY+d)

Formato binario:

1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

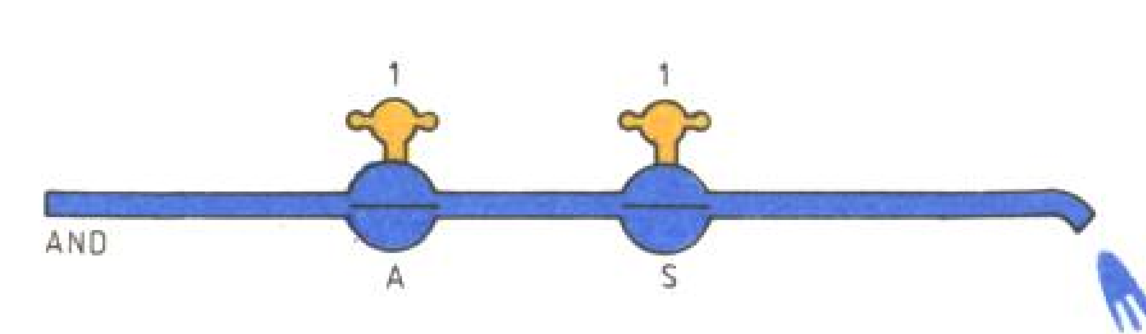
Indicadores: ver tabla

AND s

Se realiza la operación lógica AND, bit a bit, entre el operando s y el contenido del registro A, en el cual queda el resultado.

Tabla de verdad de la función AND

A	AND	s	=	A
0		0		0
0		1		0
1		0		0
1		1		1



Instr.	Hex.	Dec.
AND A	A7	167
AND B	A0	160
AND C	A1	161
AND D	A2	162
AND E	A3	163
AND H	A4	164
AND L	A5	165
AND n	E6,n	230,n
AND (HL)	A6	166
AND (IX+d)	DD,A6,d	221,166,d
AND (IY+d)	FD,A6,d	253,166,d

AND r

Mnemónico: AND

Operando: r

Formato binario:

1	0	1	0	0	r	r	r

Ciclos: 1
Estados: 4

Indicadores: ver tabla

AND n

Mnemónico: AND

Operando: n

Formato binario:

11100110
nnnnnnnn

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

AND (HL)

Mnemónico: AND

Operando: (HL)

Formato binario:

10100110

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

AND (IX+d)

Mnemónico: AND

Operando: (IX+d)

Formato binario:

11011101
10100110
dddddddd

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

AND (IY+d)

Mnemónico: AND

Operando: (IY+d)

Formato binario:

11111101
10100110
dddddddd

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1
P/V	a 1 si hay paridad (par)
N	a 0
C	a 0

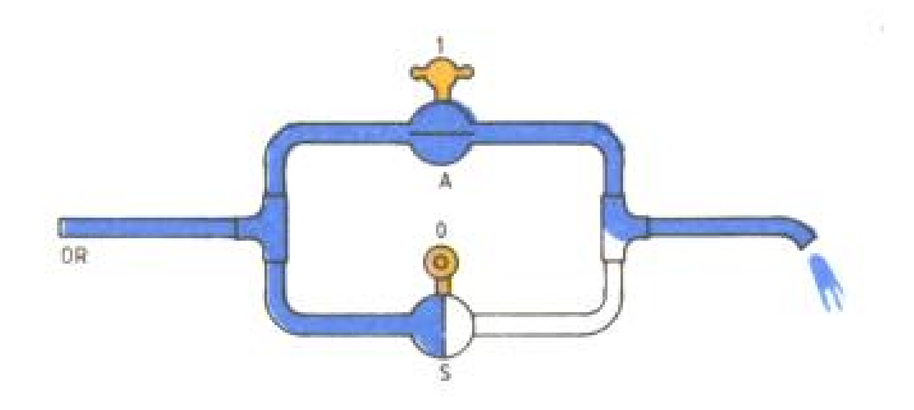
OR s

Se realiza la operación lógica OR, bit a bit, entre el operando s y el contenido del registro A, en el cual queda el resultado.

Tabla de verdad de la función OR

A	OR	s	=	A
0		0		0
0		1		1
1		0		1
1		1		1

Instr.	Hex.	Dec.
OR A	B7	183
OR B	B0	176
OR C	B1	177
OR D	B2	178
OR E	B3	179
OR H	B4	180
OR L	B5	181
OR n	F6,n	246,n
OR (HL)	B6	182
OR (IX+d)	DD,B6,d	221,182,d
OR (IY+d)	FD,B6,d	253,182,d



OR r

Mnemónico: OR

Operando: r

Formato binario:

1	0	1	1	0	r	r	r

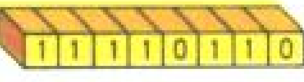
Ciclos: 1
Estados: 4

Indicadores: ver tabla

OR n

Mnemónico: OR **Operando:** n

Formato binario:





Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

OR (HL)

Mnemónico: OR **Operando:** (HL)

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

OR (IX+d)

Mnemónico: OR **Operando:** (IX+d)

Formato binario:

1 1 0 1 1 1 0 1
1 0 1 1 0 1 1 0
d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

OR (IY+d)

Mnemónico: OR **Operando:** (IY+d)

Formato binario:

1 1 1 1 1 1 0 1
1 0 1 1 0 1 1 0
d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	a 0

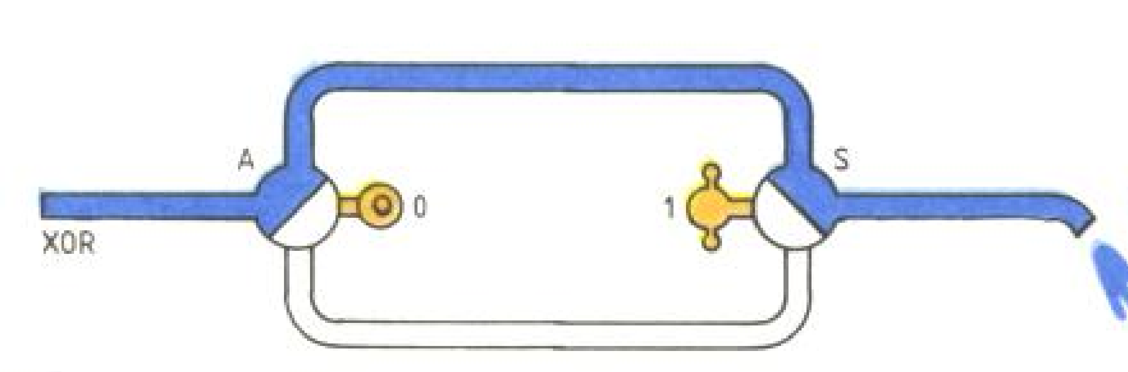
XOR s

Se realiza la operación lógica XOR, bit a bit, entre el operando s y el contenido del registro A, en el cual queda el resultado.

Tabla de verdad de la función XOR

A	XOR	s	=	A
0		0		0
0		1		1
1		0		1
1		1		0

Instr.	Hex.	Dec.
XOR A	AF	175
XOR B	A8	168
XOR C	A9	169
XOR D	AA	170
XOR E	AB	171
XOR H	AC	172
XOR L	AD	173
XOR n	EE,n	238,n
XOR (HL)	AE	174
XOR (IX+d)	DD,AE,d	221,174,d
XOR (IY+d)	FD,AE,d	253,174,d



XOR r

Mnemónico: XOR

Operando: r

Formato binario:

1	0	1	0	1	r	r	r

Ciclos: 1
Estados: 4

Indicadores: ver tabla

XOR n

Mnemónico: XOR **Operando:** n

Formato binario:

1 1 1 0 1 1 1 0

n n n n n n n n

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

XOR (HL)

Mnemónico: XOR **Operando:** (HL)

Formato binario:

1 0 1 0 1 1 1 0

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

XOR (IX+d)

Mnemónico: XOR **Operando:** (IX+d)

Formato binario:

1 1 0 1 1 1 0 1

1 0 1 0 1 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

XOR (IY+d)

Mnemónico: XOR **Operando:** (IY+d)

Formato binario:

1 1 1 1 1 1 0 1

1 0 1 0 1 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

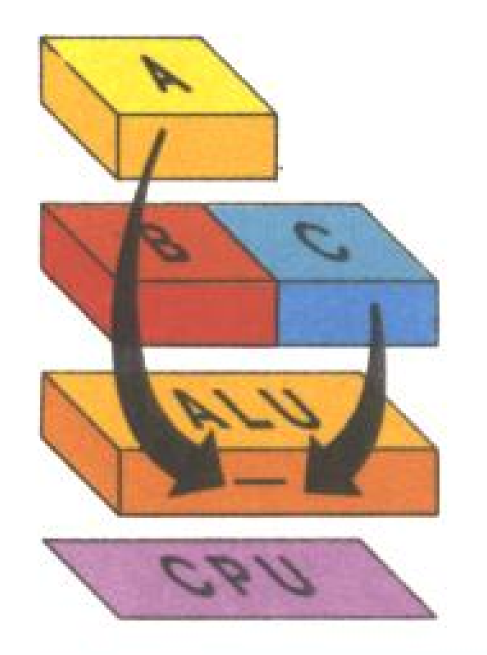
Tabla de indicadores:

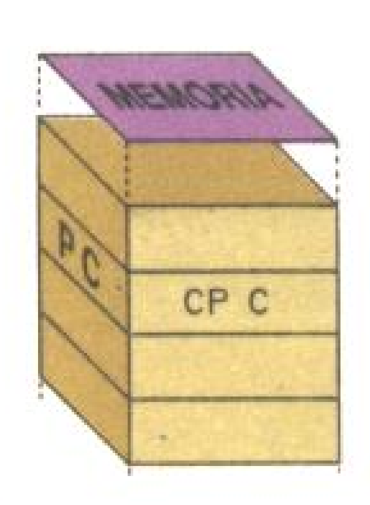
S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	a 0

CP s

El operando "s" de 8 bits es comparado con el contenido del registro A, y el resultado queda plasmado en los indicadores de condición.

La comparación equivaldría a restar al contenido del registro A el operando s, alterando sólo los indicadores de condición.





Instr.	Hex.	Dec.
CP A	BF	191
CP B	B8	184
CP C	B9	185
CP D	BA	186
CP E	BB	187
CP H	BC	188
CP L	BD	189
CP n	FE,n	254,n
CP (HL)	BE	190
CP (IX + d)	DD,BE,d	221,190,d
CP (IY + d)	FD,BE,d	223,190,d

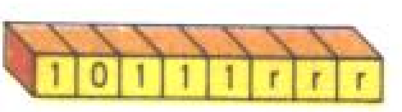
Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1 si hay acarreo del bit 3
P/V	a 1 si hay desbordamiento
N	a 1
C	a 1 si hay acarreo

CP r

Mnemónico: CP | **Operando:** r

Formato binario:



Ciclos: 1
Estados: 4

Indicadores: ver tabla

CP n

Mnemónico: CP | **Operando:** n

Formato binario:





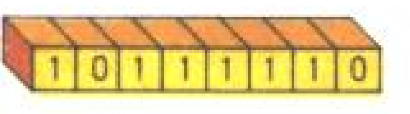
Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

CP (HL)

Mnemónico: CP | **Operando:** (HL)

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

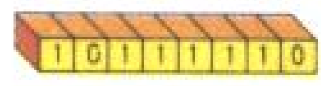
Indicadores: ver tabla

CP (IX + d)

Mnemónico: CP | **Operando:** (IX + d)

Formato binario:





d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

CP (IY + d)

Mnemónico: CP | **Operando:** (IY + d)

Formato binario:

1 1 1 1 1 1 0 1

1 0 1 1 1 1 1 0

d d d d d d d d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

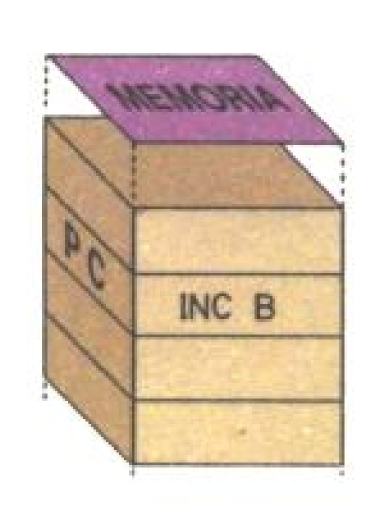
INC m

El operando "m" de 8 bits es incrementado en la unidad.

Puede ser cualquier registro r, o el contenido de la dirección de memoria especificada bien por el contenido del par HL, bien por la suma del contenido del par IX (o IY) y el desplazamiento d (de es un número de 8 bits en complemento a 2).

Instr.	Hex.	Dec.
INC A	3C	60
INC B	04	4
INC C	0C	12
INC D	14	20
INC E	1C	28
INC H	24	36
INC L	2C	44
INC (HL)	34	52
INC (IX + d)	DD,34,d	221,60,d
INC (IY + d)	FD,34,d	223,60,d





INC r

Mnemónico: INC

Operando: r

Formato binario:

0 0 r r r 1 0 0

Ciclos: 1
Estados: 4

Indicadores: ver tabla

INC (HL)

Mnemónico: INC

Operando: (HL)

Formato binario:

0	0	1	1	0	1	0	0

Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

INC (IX + d)

Mnemónico: INC

Operando: (IX + d)

Formato binario:

1	1	0	1	1	1	0	1
0	0	1	1	0	1	0	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

INC (IY + d)

Mnemónico: INC

Operando: (IY + d)

Formato binario:

1	1	1	1	1	1	0	1
0	0	1	1	0	1	0	0
d	d	d	d	d	d	d	d

Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

Tabla de indicadores:

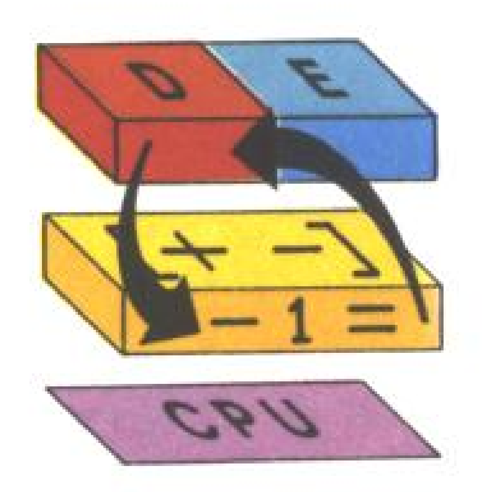
S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1 si hay acarreo del bit 3
P/V	a 1 si m contenía 7FH
N	a 0
C	no afectado

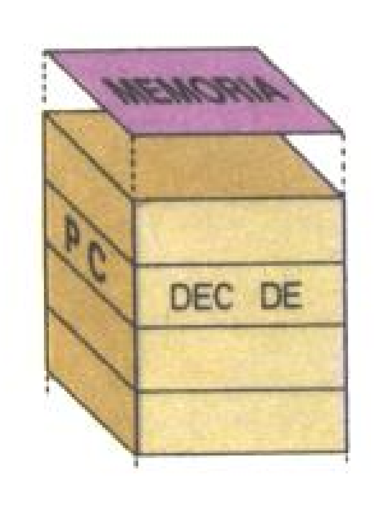
DEC m

El operando "m" de 8 bits es decrementado en la unidad.

Puede ser cualquier registro r, o el contenido de la dirección de memoria especificada bien por el contenido del par HL, bien por la suma del contenido del par IX (o IY) y el desplazamiento d (de es un número de 8 bits en complemento a 2).

Instr.	Hex.	Dec.
DEC A	3D	61
DEC B	05	5
DEC C	0D	13
DEC D	15	21
DEC E	1D	29
DEC H	25	37
DEC L	2D	45
DEC (HL)	35	53
DEC (IX + d)	DD,35,d	221,61,d
DEC (IY + d)	FD,35,d	223,61,d





DEC r

Mnemónico: DEC

Operando: r

Formato binario:

0	0	r	r	r	1	0	1

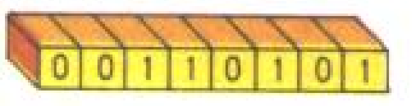
Ciclos: 1
Estados: 4

Indicadores: ver tabla

DEC (HL)

Mnemónico: DEC **Operando:** (HL)

Formato binario:



Ciclos: 2
Estados: 7 (4,3)

Indicadores: ver tabla

DEC (IY + d)

Mnemónico: DEC **Operando:** (IY + d)

Formato binario:





d d d d d d d d

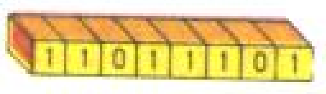
Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

DEC (IX + d)

Mnemónico: DEC **Operandos:** (IX + d)

Formato binario:



0 0 1 1 0 1 0 1



Ciclos: 5
Estados: 19 (4,4,3,5,3)

Indicadores: ver tabla

Tabla de indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 1 si hay acarreo del bit 3
P/V	a 1 si m contenía 80H
N	a 1
C	no afectado

ADD HL,ss

El contenido de 16 bits del par especificado por el operando ss, es sumado al contenido de 16 bits del par HL y el resultado queda en este último.

Mnemónico: ADD

Operando: HL,ss

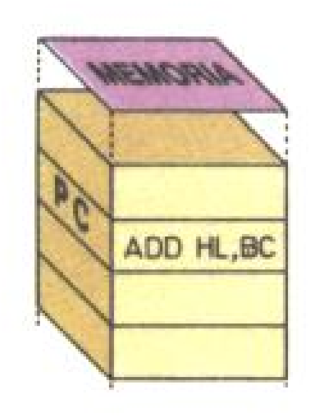
Formato binario:



Ciclos: 3
Estados: 11 (4,3,3)

Indicadores: ver tabla





Instr.	Hex.	Dec.
ADD HL,BC	09	9
ADD HL,DE	19	25
ADD HL,HL	29	41
ADD HL,SP	39	57
ADD IX,BC	DD,09	221,9
ADD IX,DE	DD,19	221,25
ADD IX,IX	DD,29	221,41
ADD IX,SP	DD,39	221,57
ADD IY,BC	FD,09	253,9
ADD IY,DE	FD,19	253,25
ADD IY,IY	FD,29	253,41
ADD IY,SP	FD,39	253,57

Tabla de indicadores:

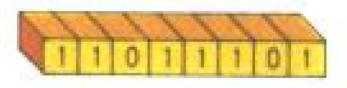
S,Z,P/V	no afectados
H	Si hay acarreo del bit 11
N	a 0
C	Si hay acarreo del bit 15

ADD IX,pp

El contenido de 16 bits del par especificado por el operando pp, es sumado al contenido de 16 bits del par IX, y el resultado queda en este último.

Mnemónico: ADD **Operandos:** IX,pp

Formato binario:

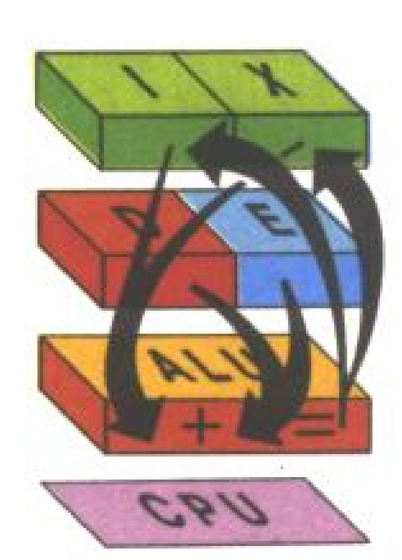


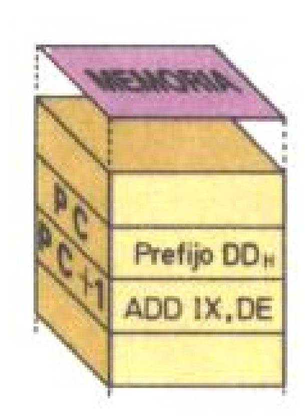


0 0 p p 1 0 0 1

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla



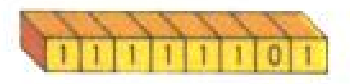


ADD IY,rr

El contenido de 16 bits del par especificado por el operando rr, es sumado al contenido de 16 bits del par IY, y el resultado queda en este último.

Mnemónico: ADD **Operandos:** IY,rr

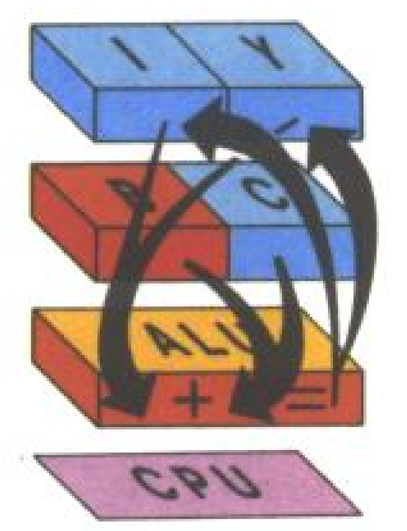
Formato binario:

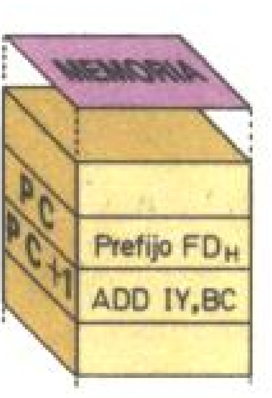




Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla





ADC HL,ss

El contenido de 16 bits del par especificado por el operando ss y el indicador C de acarreo son sumados al contenido de 16 bits del par HL, y el resultado queda en este último.

Mnemónico: ADC **Operandos:** HL,ss

Formato binario:

1 1 1 0 1 1 0 1

0 1 s s 1 0 1 0

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores:

S	a 1 si es negativo	P/V	a 1 si desborda
Z	a 1 si es cero	N	a 0
H	acarreo bit 11	C	acarreo bit 15

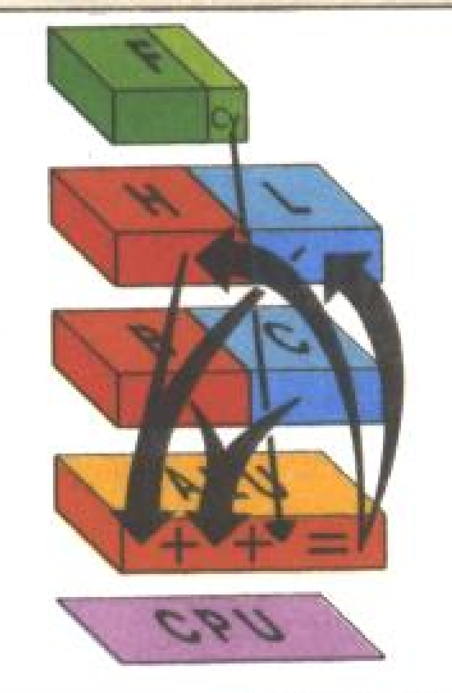
Instr.	Hex.	Dec.
ADC HL,BC	ED,4A	237,74
ADC HL,DE	ED,5A	237,90
ADC HL,HL	ED,6A	237,106
ADC HL,SP	ED,7A	237,122
SBC HL,BC	ED,42	237,66
SBC HL,DE	ED,52	237,82
SBC HL,HL	ED,62	237,98
SBC HL,SP	ED,72	237,114

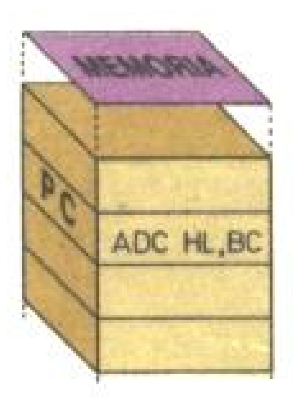
Ejemplo:

Si el par HL contiene 3333H, el indicador C está activado (contiene 1) y el par BC contiene 4326H, después de ejecutar la instrucción

ADC HL,BC

resultará que el par HL contiene 765AH.





SBC HL,ss

El contenido de 16 bits del par especificado por el operando ss y el indicador C de acarreo son restados al contenido de 16 bits del par HL, y el resultado queda en este último.

Mnemónico: SBC **Operandos:** HL,ss

Formato binario:

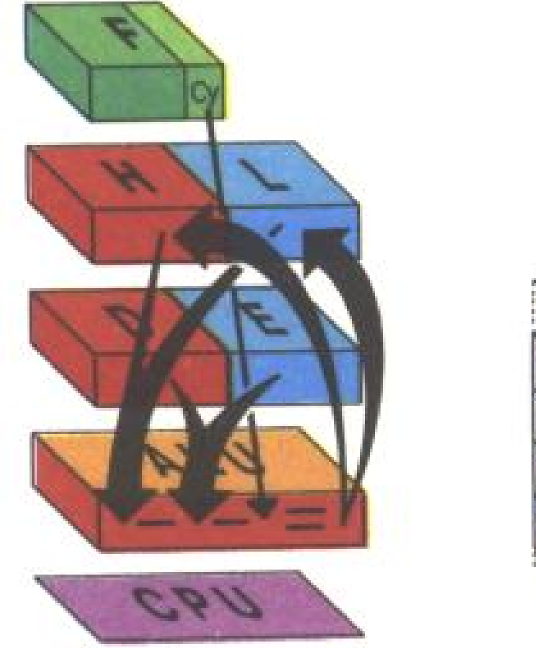
11101101

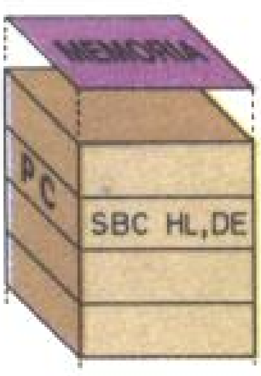
01ss0010

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores:

S	a 1 si es negativo	P/V	a 1 si desborda
Z	a 1 si es cero	N	a 1
H	acarreo bit 11	C	acarreo bit 15





Ejemplo:

Si el par HL contiene 8888H, el indicador C está activado (contiene 1) y el par DE contiene 2222H, después de ejecutar la instrucción

SBC HL,DE

el par HL contendrá 6665H.

El operando ss puede ser cualquiera de los pares según la siguiente codificación:

BC 00
DE 01
HL 10
SP 11

INC ss

El contenido de 16 bits especificado por el operando ss, es incrementado en la unidad.

Este puede ser cualquiera de los pares BC, DE, HL o SP.

Mnemónico: INC **Operando:** ss

Formato binario:

0	0	s	s	0	0	1	1

Ciclos: 1
Estados: 6

Indicadores: ninguno

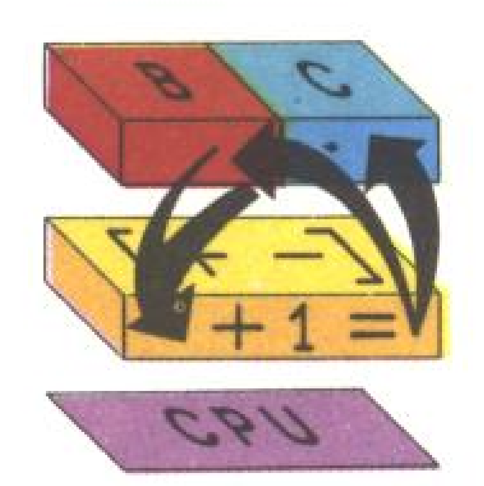
Instr.	Hex.	Dec.
INC BC	03	3
INC DE	13	19
INC HL	23	35
INC SP	33	51
INC IX	DD,23	221,35
INC IY	FD,23	253,35

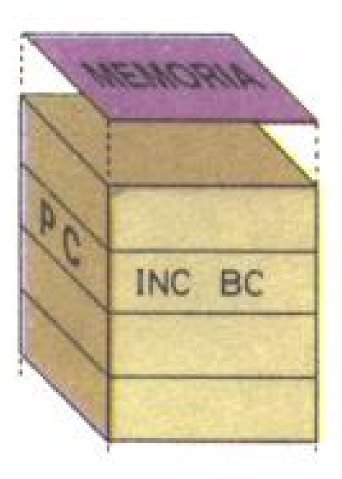
Ejemplo:

Si el par BC contiene 10FFH, después de ejecutar la instrucción

INC BC

resultará que éste contiene 1100H, puesto que la incrementación se realiza en el rango completo de 16 bits.





INC IX

El contenido de 16 bits del par IX es incrementado en la unidad.

Mnemónico: INC **Operando:** IX

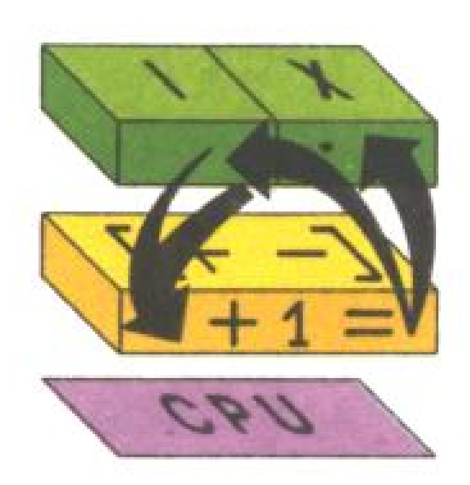
Formato binario:

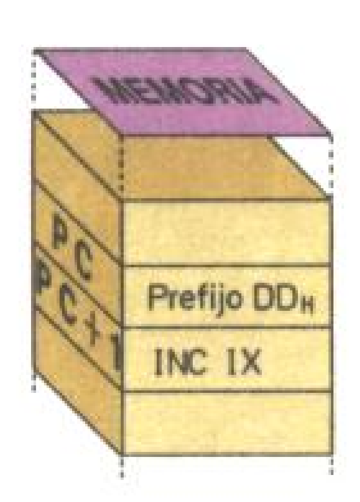
1 1 0 1 1 1 0 1



Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





INC IY

El contenido de 16 bits del par IY es incrementado en la unidad.

Mnemónico: INC **Operando:** IY

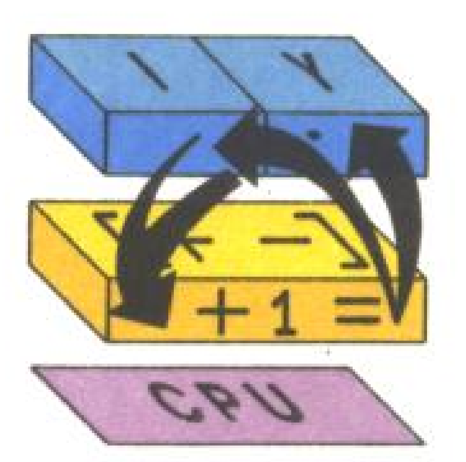
Formato binario:

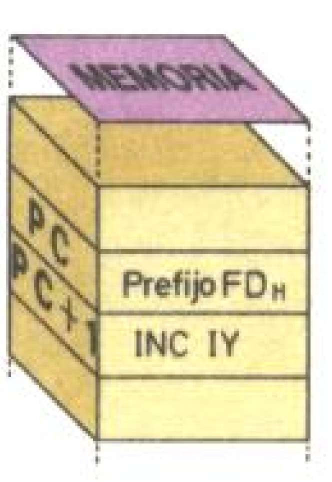




Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





DEC ss

El contenido de 16 bits especificado por el operando ss, es decrementado en la unidad.

Este puede ser cualquiera de los pares BC, DE, HL o SP.

Instr.	Hex.	Dec.
DEC BC	0B	11
DEC DE	1B	27
DEC HL	2B	43
DEC SP	3B	59
DEC IX	DD,2B	221,43
DEC IY	FD,2B	253,43

Mnemónico: DEC **Operando:** ss

Formato binario:

0	0	s	s	1	0	1	1

Ciclos: 1
Estados: 6

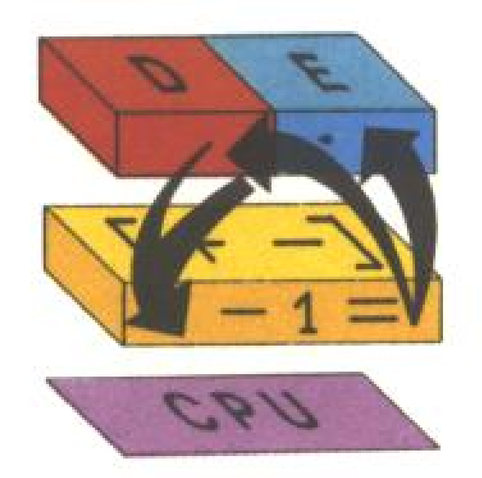
Indicadores: ninguno

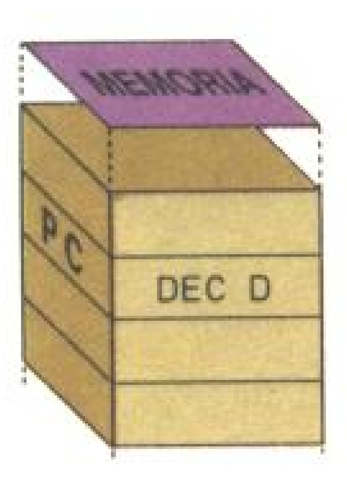
Ejemplo:

Si el par DE contiene 3000H, después de ejecutar la instrucción

DEC DE

resultará que éste contiene 2FFFH, puesto que la decrementación se realiza en el rango completo de 16 bits.





DEC IX

El contenido de 16 bits del par IX es decrementado en la unidad.

Mnemónico: DEC **Operando:** IX

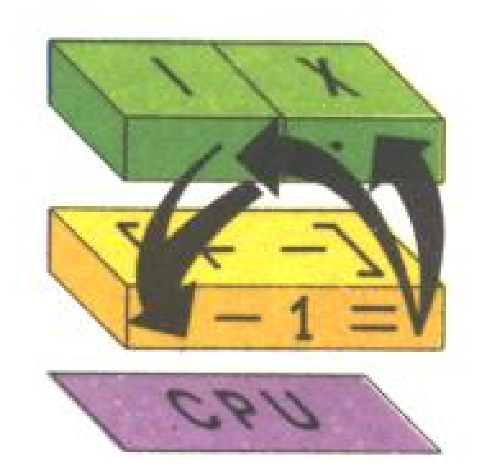
Formato binario:

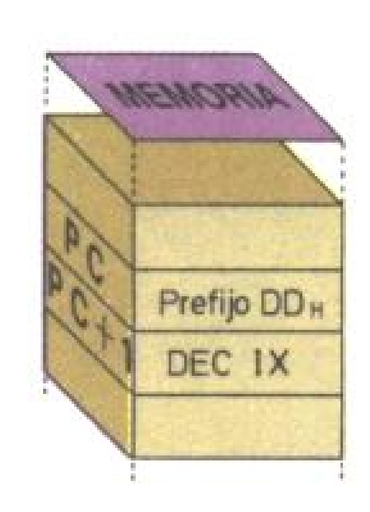
1 1 0 1 1 1 0 1

0 0 1 0 1 0 1 1

Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





DEC IY

El contenido de 16 bits del par IY es decrementado en la unidad.

Mnemónico: DEC **Operando:** IY

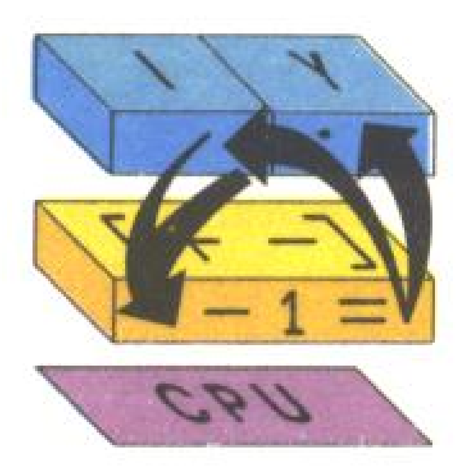
Formato binario:

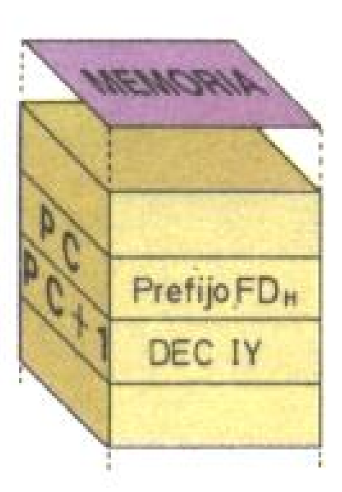
1 1 1 1 1 1 0 1

0 0 1 0 1 0 1 1

Ciclos: 2
Estados: 10 (4,6)

Indicadores: ninguno





PUSH qq   PUSH IX   PUSH IY

PUSH qq

El contenido de 16 bits especificado por el operando qq, es almacenado en la pila de máquina. Primero se decrementa el par SP, y en la dirección que éste contenga se carga la parte alta del operando qq; se decrementa nuevamente el par SP y en la dirección que contenga se carga la parte baja del operando qq.

Instr.	Hex.	Dec.
PUSH BC	C5	197
PUSH DE	D5	213
PUSH HL	E5	229
PUSH AF	F5	245
PUSH IX	DD,E5	221,229
PUSH IY	FD,E5	253,229

Mnemónico: PUSH   **Operando:** qq

Formato binario:

1 1 q q 0 1 0 1

Ciclos: 3
Estados: 11 (5,3,3)

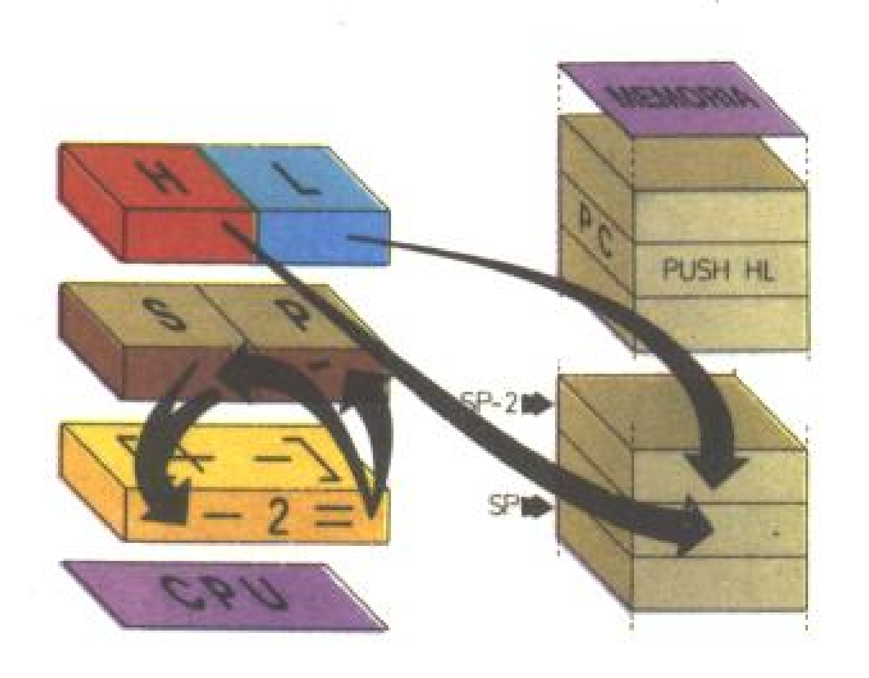
Indicadores: ninguno

Ejemplo:

Si el par HL contiene 1020H y el par SP contiene 3040H, después de ejecutar la instrucción

PUSH HL

resultará que el par SP contiene 303EH, que en la dirección 303FH contiene 10H, y la dirección 303EH contiene 20H.



PUSH IX

El contenido de 16 bits del par IX es almacenado en la pila de máquina.

Mnemónico: PUSH **Operando:** IX

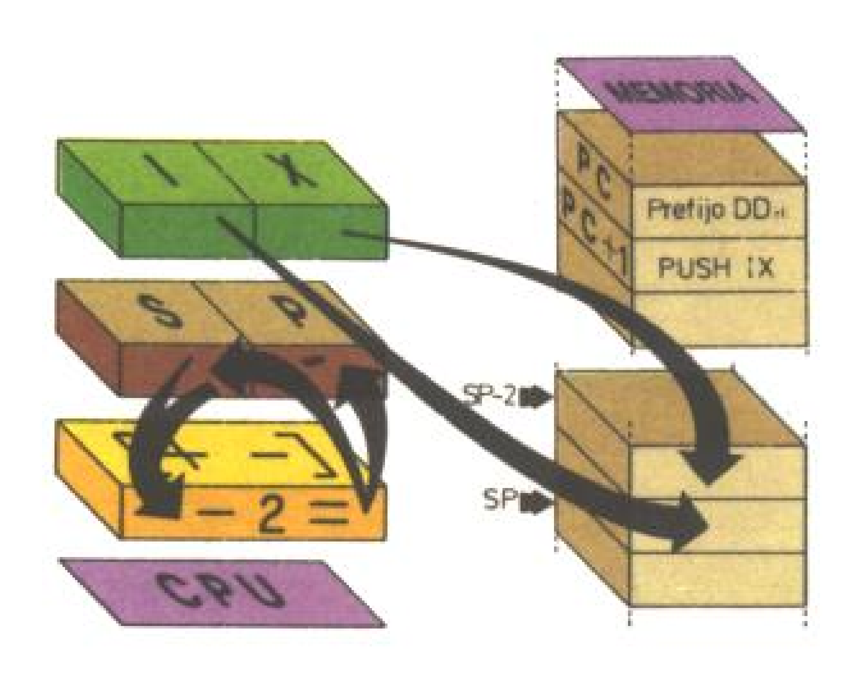
Formato binario:





Ciclos: 4
Estados: 15 (4,5,3,3)

Indicadores: ninguno





PUSH IY

El contenido de 16 bits del par IY es almacenado en la pila de máquina.

Mnemónico: PUSH **Operando:** IY

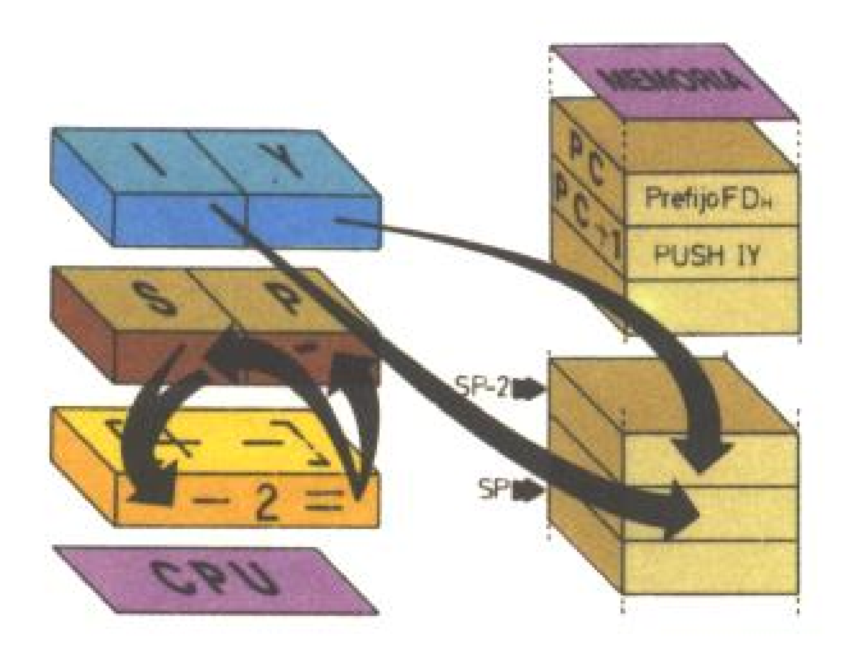
Formato binario:

1 1 1 1 1 1 0 1



Ciclos: 4
Estados: 15 (4,5,3,3)

Indicadores: ninguno



POP qq

El último dato de 16 bits almacenado en la pila de máquina es transferido al par especificado por el operado qq.

Primero, se carga la parte baja del par qq con el contenido de la dirección especificada por el contenido del par SP; se incrementa el par SP, se carga la parte alta del par qq de la misma manera y se vuelve a incrementar el par SP.

Mnemónico: POP   **Operando:** qq

Formato binario:

1 1 q q 0 0 0 1

Ciclos: 3
Estados: 11 (5,3,3)

Indicadores: ninguno

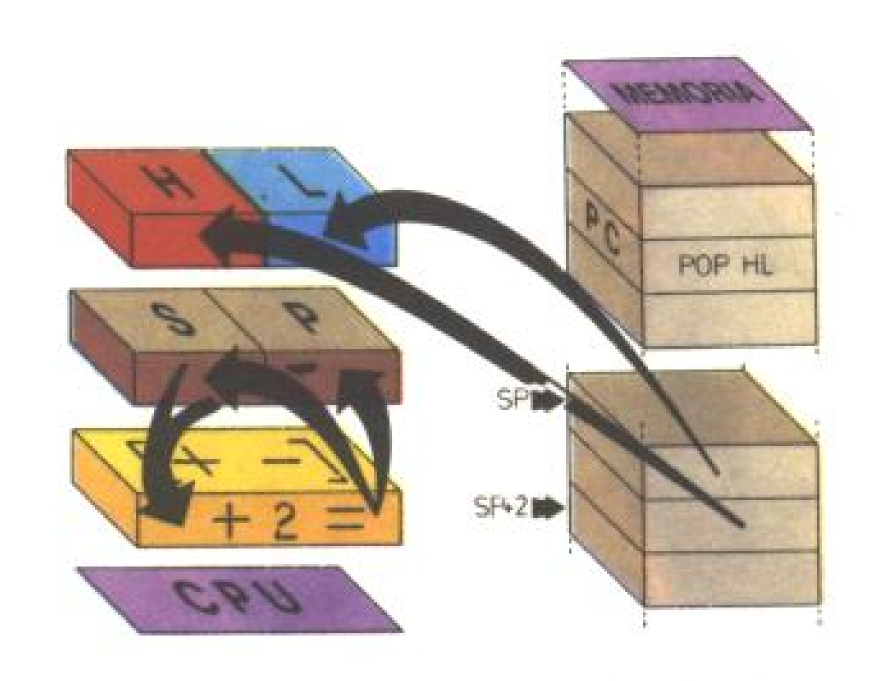
Ejemplo:

Si el par SP contiene 9000H, la dirección 9000H contiene 12H, y la dirección 9001H contiene 34H, después de ejecutar la instrucción.

POP HL

resultará que el par HL contiene 3412H y el par SP contiene 9002H.

Instr.	Hex.	Dec.
POP BC	C1	193
POP DE	D1	209
POP HL	E1	225
POP AF	F1	241
POP IX	DD,E1	221,225
POP IY	FD,E1	253,225



POP IX

El último dato de 16 bits almacenado en la pila de máquina es transferido al par IX.

Mnemónico: POP **Operando:** IX

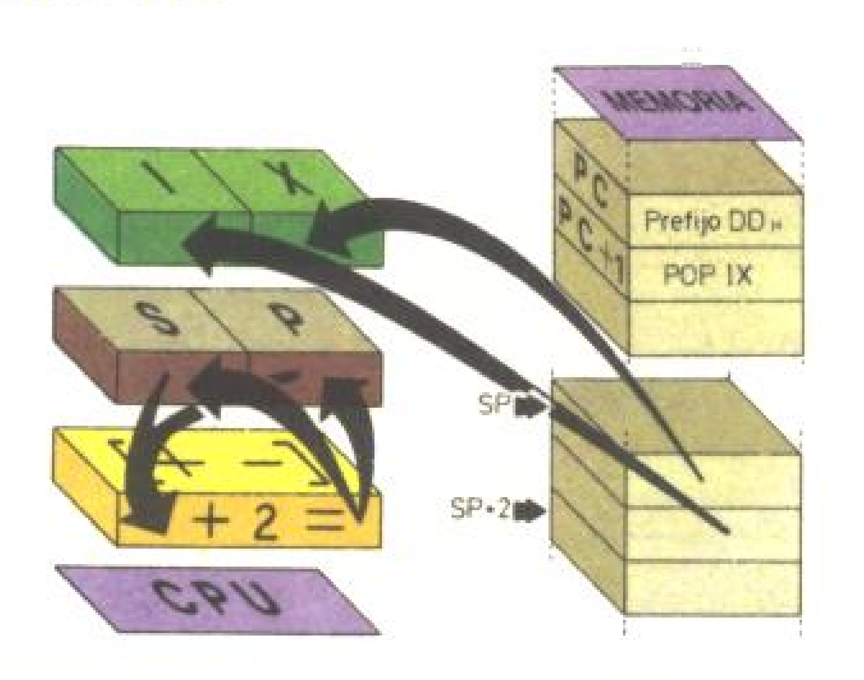
Formato binario:



Ciclos: 4
Estados: 15 (4,5,3,3)

Indicadores: ninguno





POP IY

El último dato de 16 bits almacenado en la pila de máquina es transferido al par IY.

Mnemónico: POP **Operando:** IY

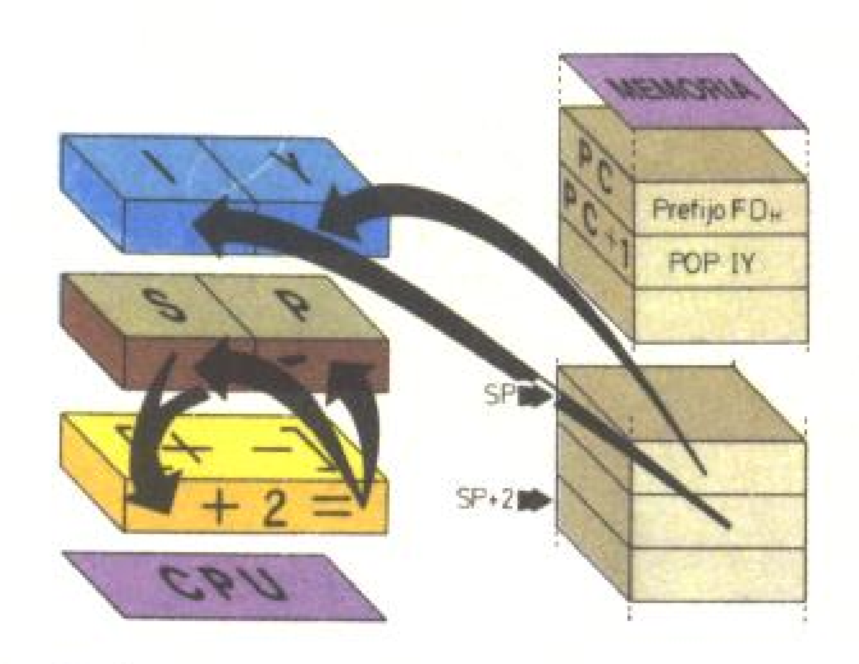
Formato binario:

1 1 1 1 1 1 0 1

1 1 1 0 0 0 0 1

Ciclos: 4
Estados: 15 (4,5,3,3)

Indicadores: ninguno



LDI

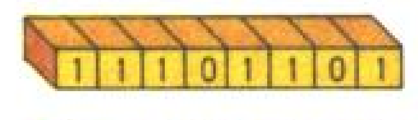
El byte que ocupa la posición de memoria especificada por el contenido del par HL es transferido a la posición especificada por el contenido del par DE, y a continuación ambos pares son incrementados.

El par BC es decrementado, lo que permite utilizarlo como contador en un bucle de LDDs sucesivos.

Instr.	Hex.	Dec.
LDI	ED,A0	237,160
LDIR	ED,B0	237,176

Mnemónico: LDI **Operandos:** no tiene

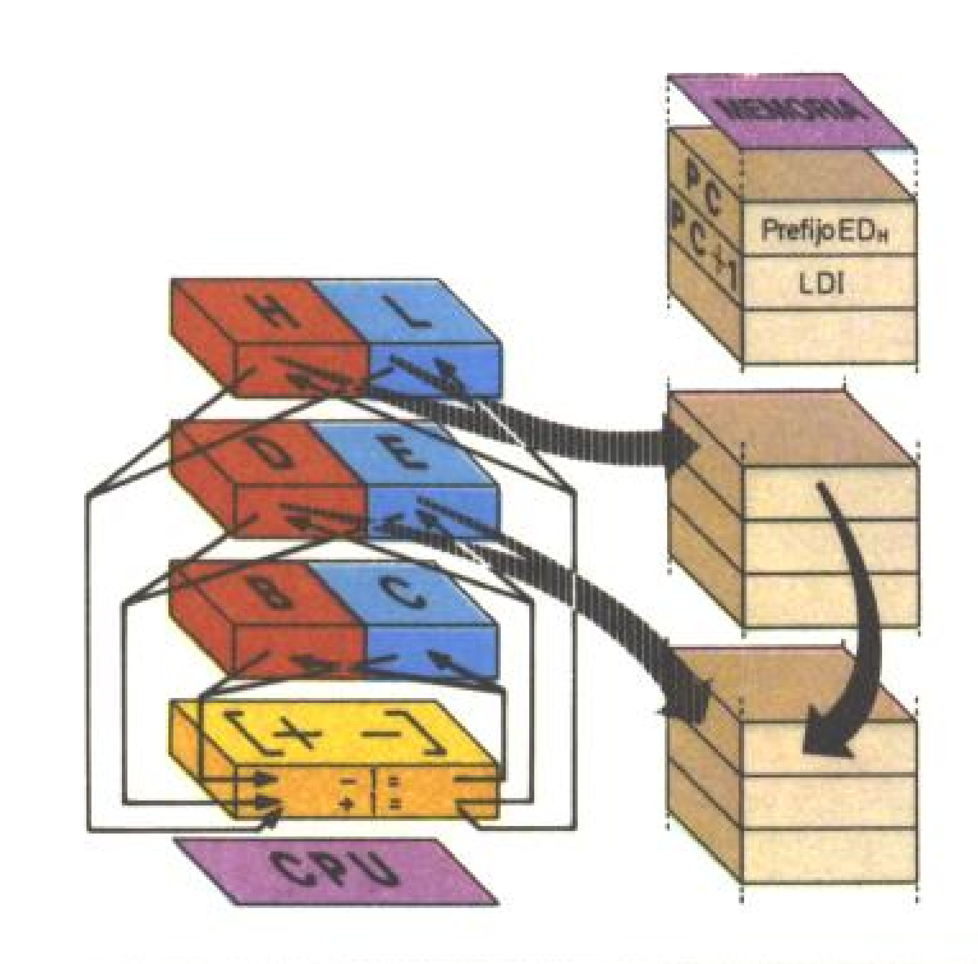
Formato binario:





Ciclos: 4
Estados: 16 (4,4,3,5)

Indicadores:
S no afectado
Z no afectado
H a 0
P/V a 0 si BC resulta 0
N a 0
C no afectado



LDIR

Se repite la secuencia LDI hasta que el par BC contiene 0, en cuyo caso termina la instrucción.

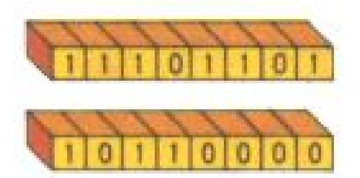
Por lo tanto, se transfiere el contenido de un bloque de memoria que comienza en la dirección especificada por el par HL, de longitud especificada por el par BC, a otro bloque de memoria que comienza en la posición especificada por el par DE.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: LDIR **Operandos:** no tiene

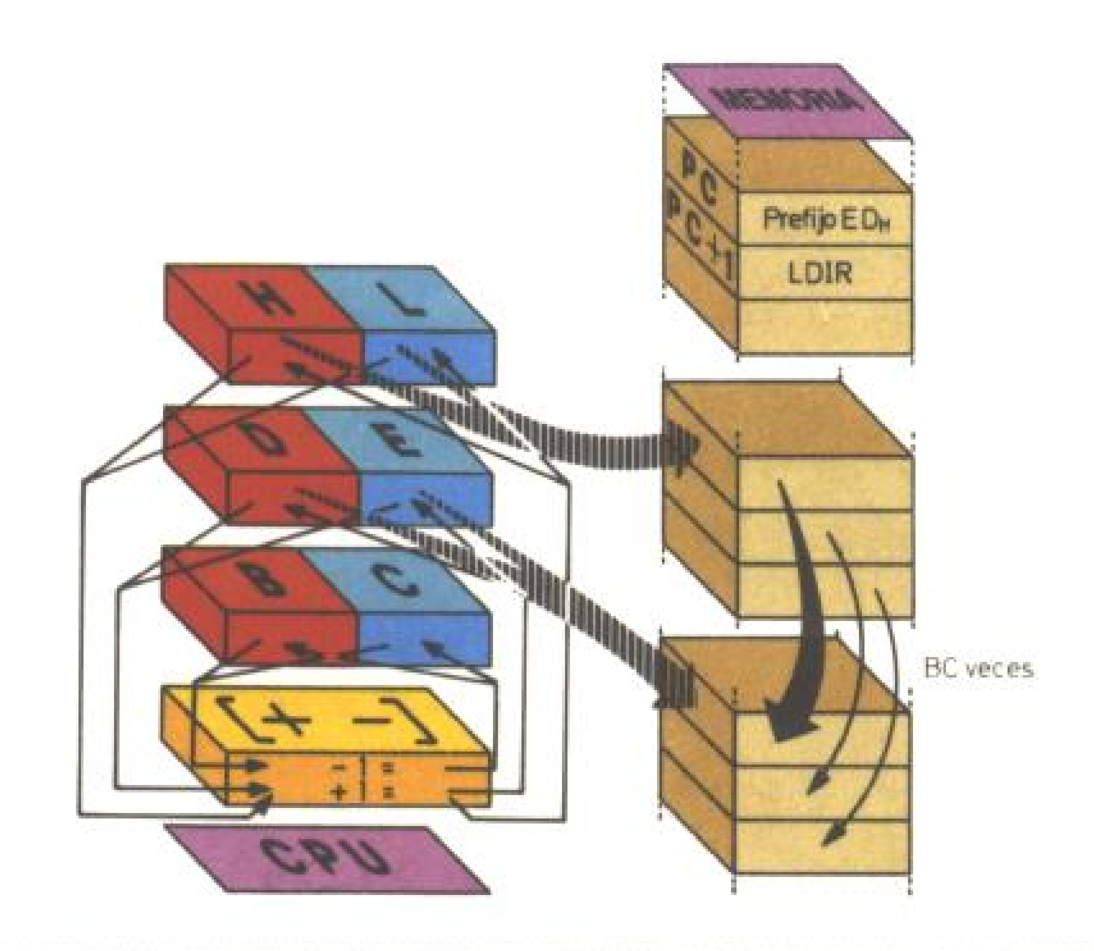
para BC < >0	para BC = 0
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,4,3,5,5)	**Estados:** 16 (4,4,3,5)

Formato binario:



Indicadores:

S no afectado	P/V a 0
Z no afectado	N a 0
H a 0	C no afectado



Instr.	Hex.	Dec.
LDD	ED,A8	237,168
LDDR	ED,B8	237,184

LDD

El byte que ocupa la posición de memoria especificada por el contenido del par HL es transferido a la posición especificada por el contenido del par DE, y a continuación ambos pares son decrementados.

El par BC también es decrementado, lo que permite utilizarlo como contador en un bucle de LDDs sucesivos.

Mnemónico: LDD

Operandos: no tiene

Formato binario:

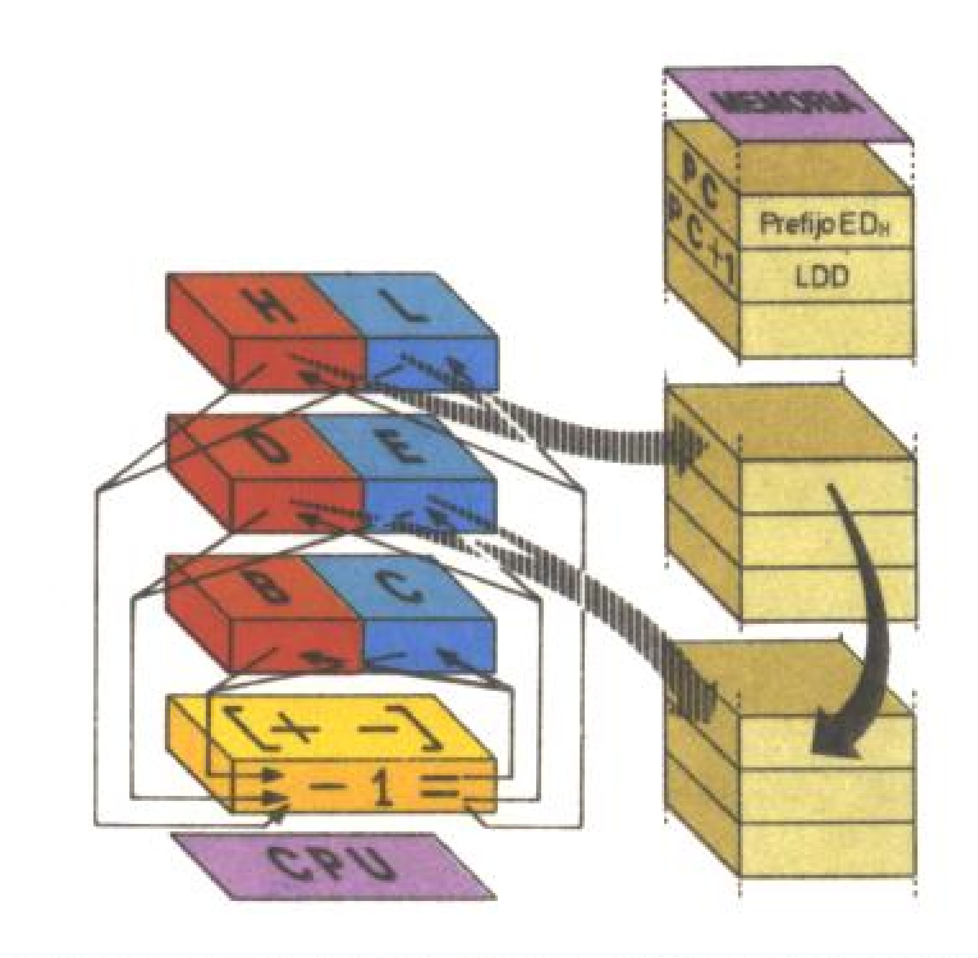
1 1 1 0 1 1 0 1

1 0 1 0 1 0 0 0



Ciclos: 4
Estados: 16 (4,4,3,5)

Indicadores:
S no afectado
Z no afectado
H a 0
P/V a 0 si BC resulta 0
N a 0
C no afectado



LDDR

Se repite la secuencia LDD hasta que el par BC contiene 0, en cuyo caso termina la instrucción.

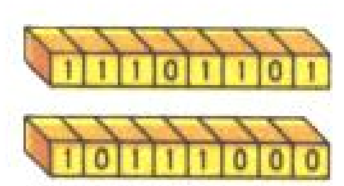
Por lo tanto, se transfiere el contenido de un bloque de memoria que termina en la dirección especificada por el par HL, de longitud especificada por el par BC, a otro bloque de memoria que termina en la posición especificada por el par DE.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: LDDR **Operandos:** no tiene

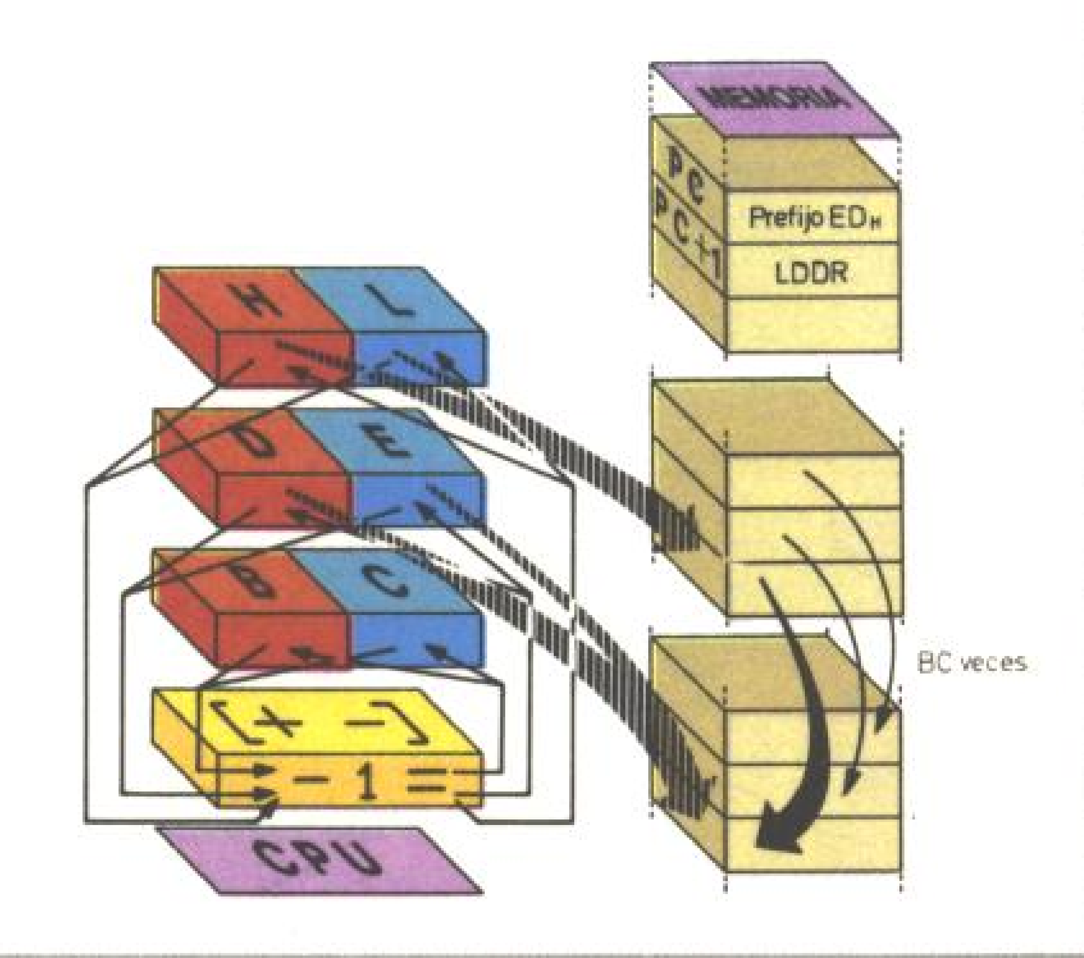
para BC < > 0	para BC = 0
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,4,3,5,5)	**Estados:** 16 (4,4,3,5)

Formato binario:



Indicadores:

S no afectado
Z no afectado
H a 0
P/V a 0
N a 0
C no afectado



CPI

El byte que ocupa la posición de memoria especificada por el contenido del par HL es comparado con el contenido del registro A.

La comparación consiste en restarle a A el contenido de (HL), sin variar éste, pero poniendo los indicadores según el resultado de la resta. El par HL es incrementado y el par BC es decrementado.

Mnemónico: CPI

Operandos: no tiene

Formato binario:

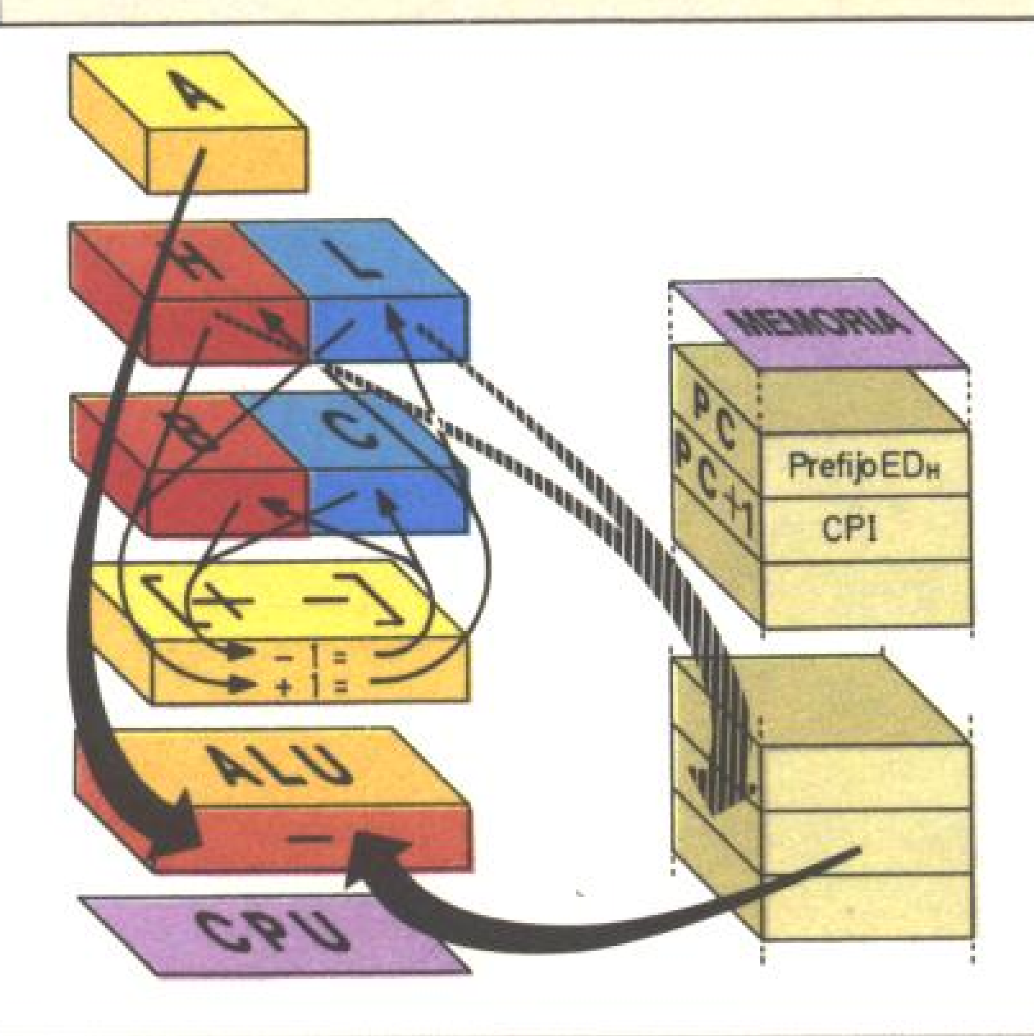




Ciclos: 4
Estados: 16 (4,4,3,5)

Indicadores:
S a 1 si es negativo
Z a 1 si A = (HL)
H acarreo del bit 3
P/V a 0 si BC resulta 0
N a 1
C no afectado

Instr.	Hex.	Dec.
CPI	ED,A1	237,161
CPIR	ED,B1	237,177



CPIR

Se repite la secuencia CPI hasta que el par BC contiene 0, o se encuentra una coincidencia entre A y (HL), y en cualquiera de ambos casos termina la instrucción.

Por lo tanto, se busca el byte contenido en el registro A, dentro de un bloque de memoria que comienza en la dirección especificada por el par HL, de longitud especificada por el par BC.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: CPIR **Operandos:** no tiene

para BC < > 0 y A < > (HL)	para BC = 0 o A = (HL)
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,4,3,5,5)	**Estados:** 16 (4,4,3,5)

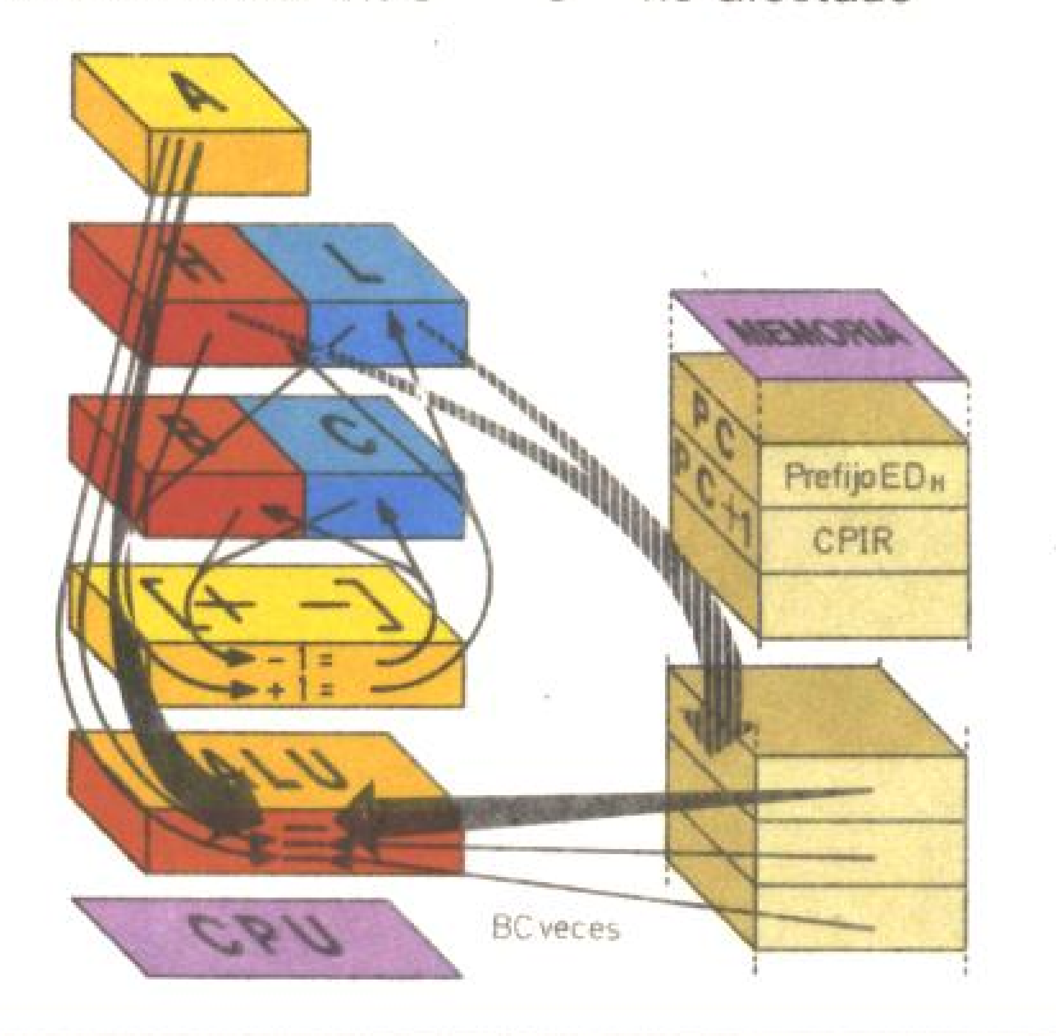
Formato binario:

1 1 1 0 1 1 0 1



Indicadores:

S	a 1 si es negativo	P/V	a 0 si BC resulta 0
Z	a 1 si A = (HL)	N	a 1
H	acarreo del bit 3	C	no afectado



CPD

El byte que ocupa la posición de memoria especificada por el contenido del par HL es comparado con el contenido del registro A.

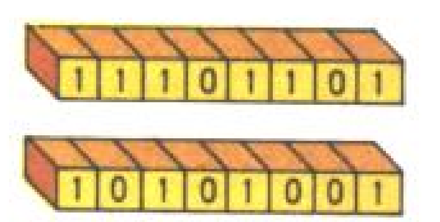
La comparación consiste en restarle a A el contenido de (HL), sin variar éste, pero poniendo los indicadores según el resultado de la resta.

El par HL, y el par BC son decrementados.

Instr.	Hex.	Dec.
CPD	ED,A9	237,169
CPDR	ED,B9	237,185

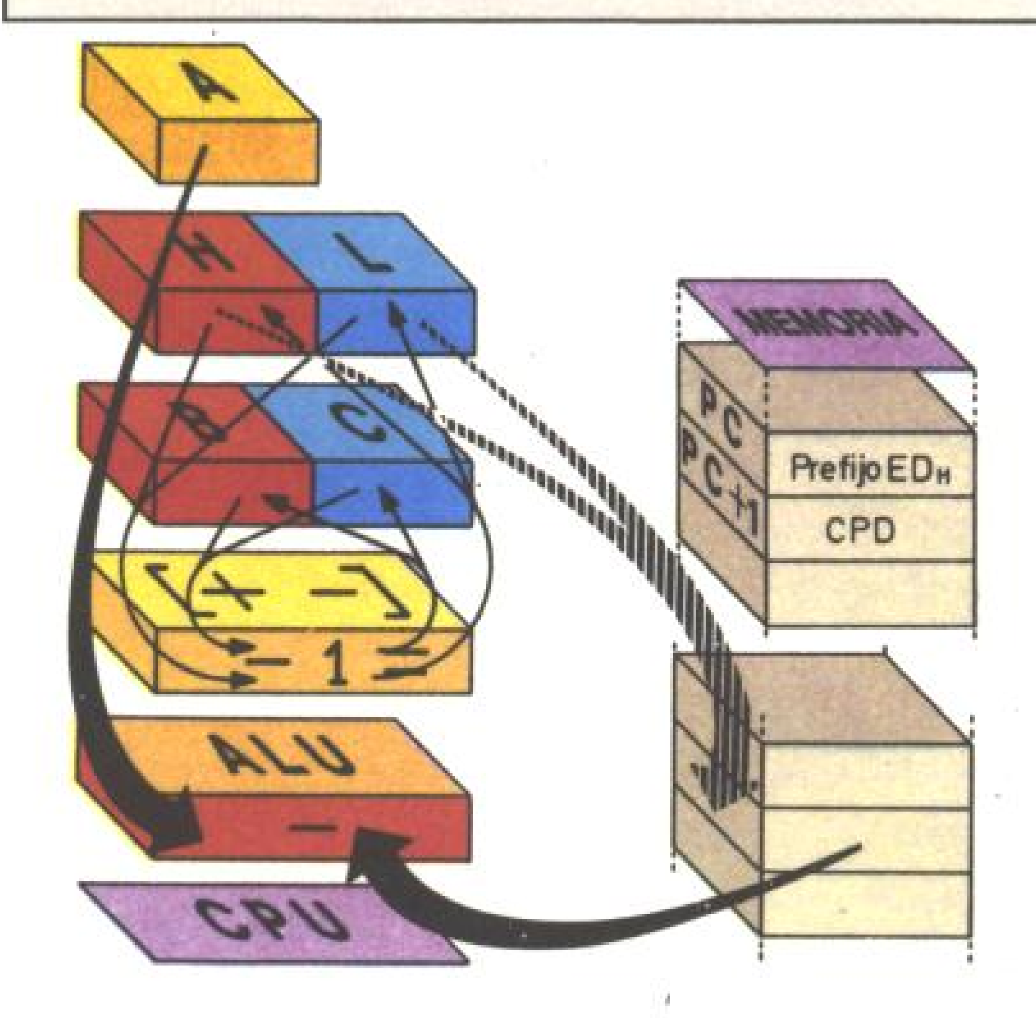
Mnemónico: CPD **Operandos:** no tiene

Formato binario:



Ciclos: 4
Estados: 16 (4,4,3,5)

Indicadores:
S a 1 si es negativo
Z a 1 si A = (HL)
H acarreo del bit 3
P/V a 0 si BC resulta 0
N a 1
C no afectado



CPDR

Se repite la secuencia CPD hasta que el par BC contiene 0, o se encuentra una coincidencia entre A y (HL), y en cualquiera de ambos casos termina la instrucción.

Por lo tanto, se busca el byte contenido en el registro A, dentro de un bloque de memoria que termina en la dirección especificada por el par HL, de longitud especificada por el par BC.

Las interrupciones son comprobadas al final de cada transferencia.

Mnemónico: CPDR **Operandos:** no tiene

para BC < > 0 y A < > (HL)	para BC = 0 o A = (HL)
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,4,3,5,5)	**Estados:** 16 (4,4,3,5)

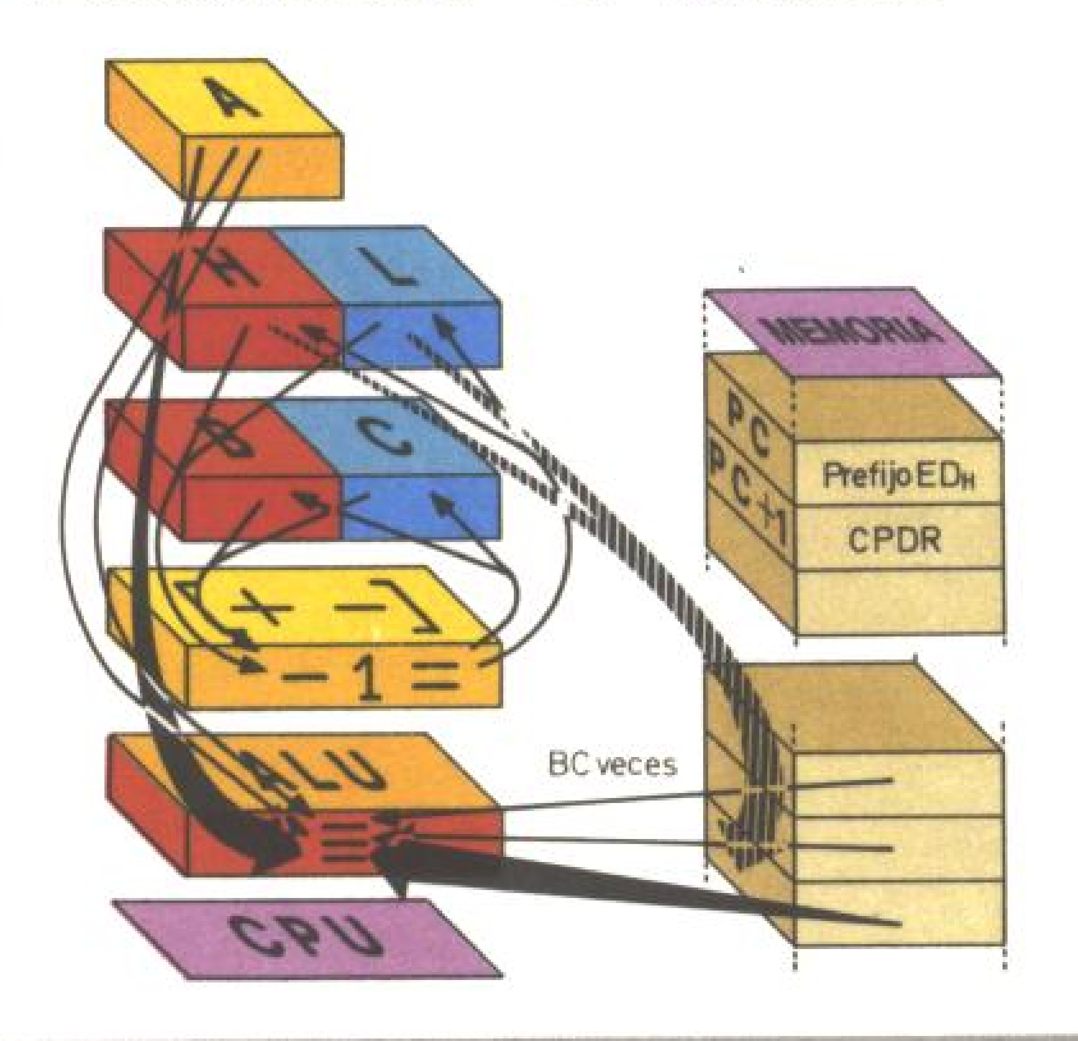
Formato binario:

1 1 1 0 1 1 0 1

1 0 1 1 1 0 0 1

Indicadores:

S a 1 si es negativo	P/V a 0 si BC resulta 0
Z a 1 si A = (HL)	N a 1
H acarreo del bit 3	C no afectado



DAA

Ajuste decimal del acumulador: El contenido del acumulador es modificado tras una suma o una resta, para que el resultado de la operación corresponda a la representación correcta de un decimal codificado en Binario (BCD).

Instr.	Hex.	Dec.
DAA	27	39
CPL	2F	47
NEG	ED,44	237,68

Mnemónico: DAA **Operandos:** no tiene

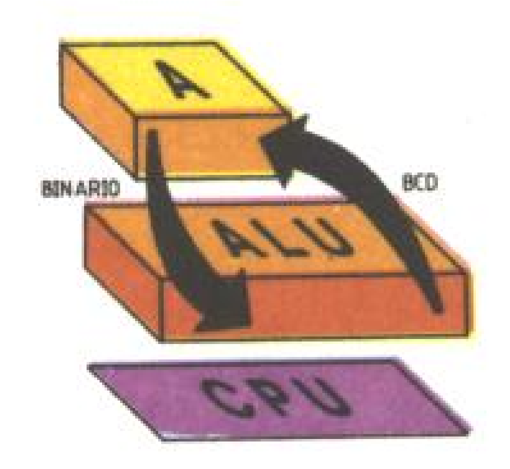
Formato binario:

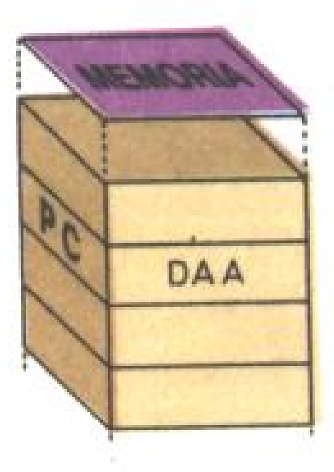
0 0 1 0 0 1 1 1

Ciclos: 1
Estados: 4

Indicadores:

S como el bit 7
Z a 1 si es cero
H si el 1.er dígito > 9
P/V a 1 si hay paridad
N no afectado
C si es mayor de 99





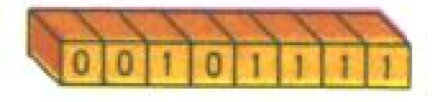
CPL

El contenido del acumulador es complementado: Los unos pasan a ser ceros y los ceros unos. (Complemento a uno).

Mnemónico: CPL

Operandos: no tiene

Formato binario:



Ciclos: 1
Estados: 4

Indicadores:

S no afectado
Z no afectado
H a 1
P/V no afectado
N a 1
C no afectado

NEG

El contenido del acumulador es restado de cero quedando el resultado en el acumulador. (Complemento a dos).

Mnemónico: NEG

Operandos: no tiene

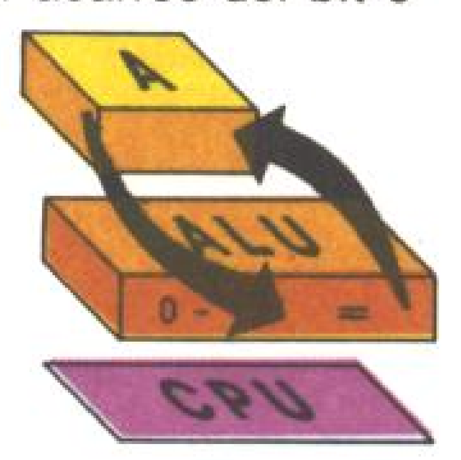
Formato binario:

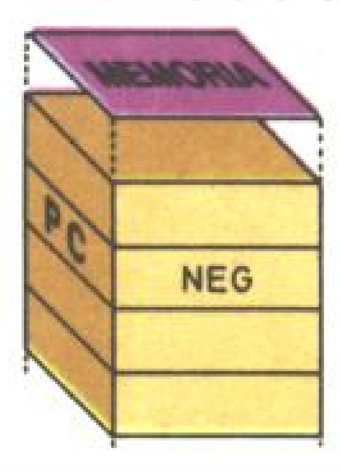


Ciclos: 1
Estados: 4

Indicadores:

S a 1 si es negativo
Z a 1 si es cero
H acarreo del bit 3
P/V a 1 si era 80H
N a 1
C a 1 si no era 00H





CCF

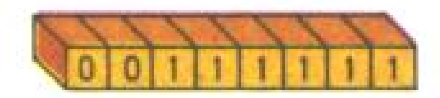
El bit indicador de acarreo (carry) del registro de banderas «F» es complementado, esto es, toma el valor 1 si anteriormente era un 0, y pasa a ser 1 en caso de que el valor inicial fuera 0.

Instr.	Hex.	Dec.
CCF	3F	63
SCF	37	55
NOP	00	0
HALT	76	118

Mnemónico: CCF

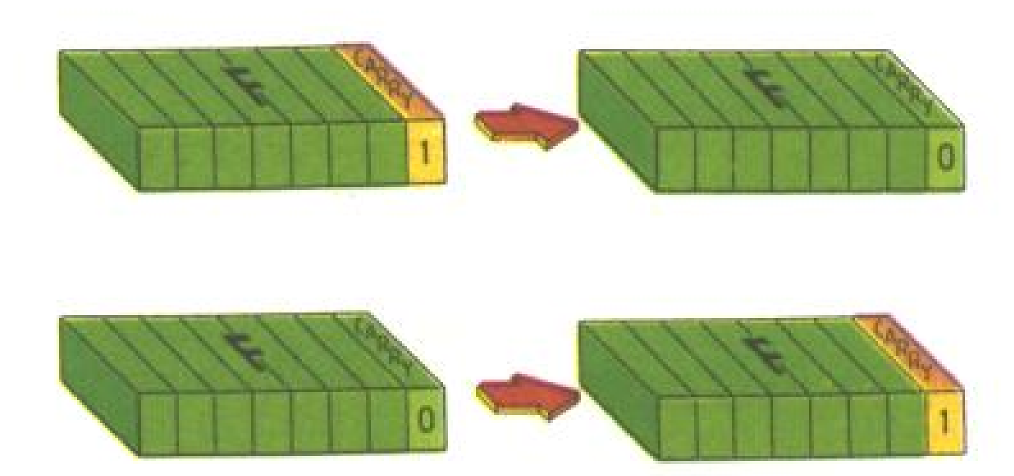
Operandos: no tiene

Formato binario:



Ciclos: 1
Estados: 4

Indicadores:
S no afectado
Z no afectado
H carry anterior
P/V no afectado
N a 1
C se invierte su valor

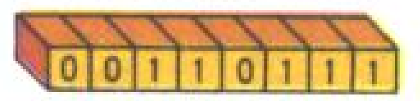


SCF

El bit indicador de acarreo (Carry) del registro «F» es puesto a uno. (Bandera alzada).

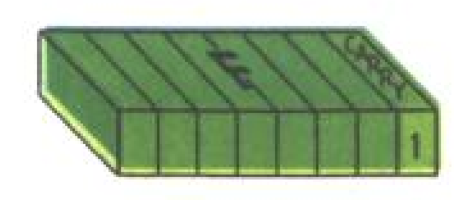
Mnemónico: SCF **Operandos:** no tiene

Formato binario:

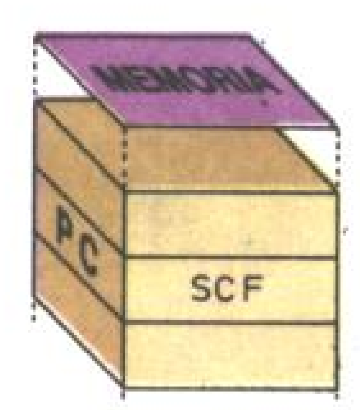


Ciclos: 1
Estados: 4

Indicadores:
S no afectado
Z no afectado
H a 0
P/V no afectado
N a 0
C a 1







NOP

La CPU no realiza ninguna operación.

Mnemónico: NEG **Operandos:** no tiene

Formato binario:

0 0 0 0 0 0 0 0

Ciclos: 1
Estados: 4

Indicadores: ninguno

HALT

La CPU se para hasta recibir una llamada de interrupción o reset.

Mnemónico: HALT **Operandos:** no tiene

Formato binario:

0 1 1 1 0 1 1 0

Ciclos: 1
Estados: 4

Indicadores: ninguno

DI

Las interrupciones enmascarables son deshabilitadas hasta que se rehabiliten mediante la instrucción EI. Son desconectados los interruptores flips-flops (IFF1 y IFF2). La CPU no podrá responder a la señal $\overline{\text{INT}}$.

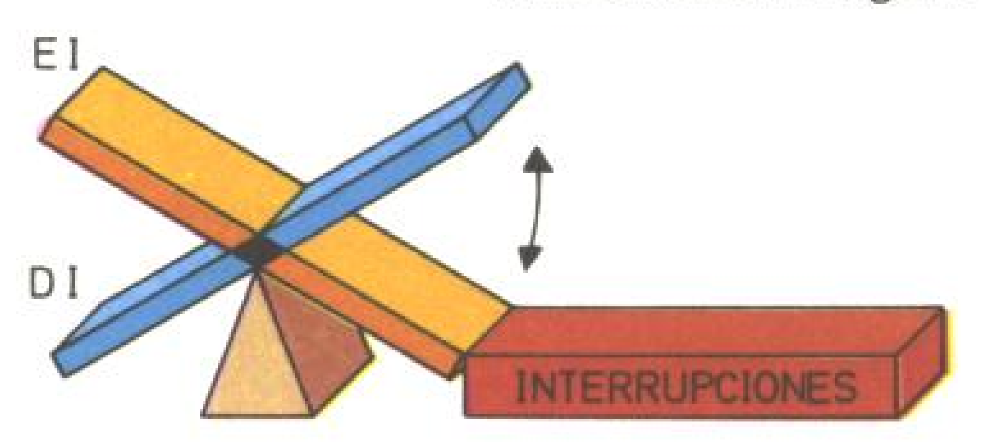
Mnemónico: DI **Operandos:** no tiene

Formato binario:

1 1 1 1 0 0 1 1

Ciclos: 1
Estados: 4

Indicadores: ninguno



Instr.	Hex.	Dec.
DI	F3	243
EI	FB	251
IM0	ED,46	237,70
IM1	ED,56	237,86
IM2	ED,5E	237,94

EI

Son habilitadas las interrupciones enmascarables al ser conectados los flips-flops (IFF1 e IFF2). Esta instrucción deshabilita las interrupciones durante su ejecución.

Mnemónico: EI **Operandos:** no tiene

Formato binario:

1 1 1 1 1 0 1 1

Ciclos: 1
Estados: 4

Indicadores: ninguno

IM0

Sitúa la CPU en el modo 0 de interrupciones enmascarables. En este modo el dispositivo de interrupciones puede insertar cualquier instrucción en el bus de datos y hacer que la CPU la ejecute continuando el programa su curso posteriormente.

Mnemónico: IM

Formato binario:





Operandos: 0

Ciclos: 2
Estados: 8 (4,4)

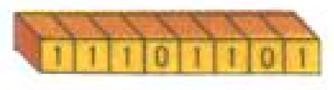
Indicadores: ninguno

IM1

Es activado el modo 1 de interrupciones. En este modo a la llamada de una interrupción enmascarable es ejecutada la instrucción RST 38H (FFH). Es el modo normal de funcionamiento del Spectrum.

Mnemónico: IM

Formato binario:





Operandos: 1

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ninguno

IM2

Modo 2 de interrupciones enmascarables. La CPU hace un CALL a la dirección de memoria contenida en la dirección determinada por el registro I (Parte alta) y el contenido del bus de datos (parte baja). El Spectrum pone FFH en el bus de datos.

Mnemónico: IM

Formato binario:





Operandos: 2

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ninguno

RLCA

Rotación circular a la izquierda del acumulador. El bit 7 además de pasar al 0 es copiado en el Carry.

Mnemónico: RLCA **Operandos:** no tiene

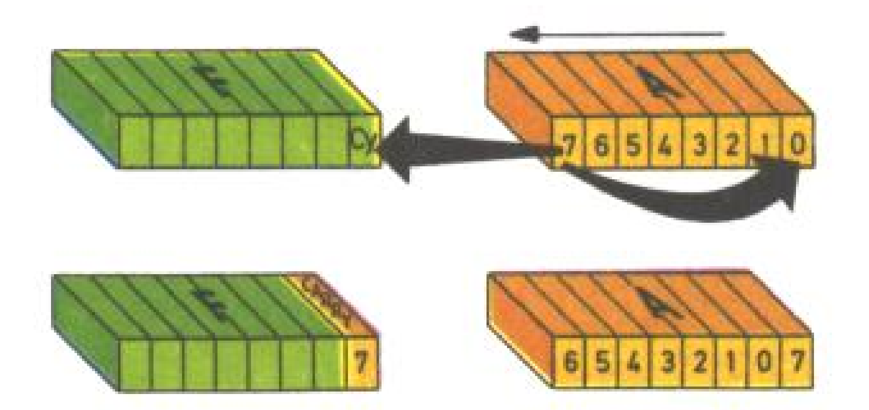
Formato binario:

0 0 0 0 0 1 1 1

Ciclos: 1
Estados: 4

Indicadores:

S no afectado
Z no afectado
H a 0
P/V no afectado
N a 0
C anterior bit 7



Instr.	Hex.	Dec.
RLCA	07	7
RLC A	CB,07	203,7
RLC B	CB,00	203,0
RLC C	CB,01	203,1
RLC D	CB,02	203,2
RLC E	CB,03	203,3
RLC H	CB,04	203,4
RLC L	CB,05	203,5
RLC (HL)	CB,06	203,6
RLC (IX + d)	DD,CB,d,06	221,203,d,6
RLC (IY + d)	FD,CB,d,06	253,203,d,6

RLC r

Rotación circular a la izquierda de un registro.

Mnemónico: RLC **Operandos:** r

Formato binario:

1 1 0 0 1 0 1 1
0 0 0 0 0 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla

RLC (HL)

Rotación circular a la izquierda del contenido de la dirección de memoria especificada por el par HL.

Mnemónico: RLC **Operandos:** (HL)

Formato binario

1	1	0	0	1	0	1	1
0	0	0	0	0	1	1	0

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla

RLC (IX + d)

Rotación circular a la izquierda del contenido de la dirección de memoria especificada por la suma del par IX más el desplazamiento d.

Mnemónico: RLC **Operandos:** (IX + d)

Formato binario:

1	1	0	1	1	1	0	1
1	1	0	0	1	0	1	1
d	d	d	d	d	d	d	d
0	0	0	0	0	1	1	0

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

RLC (IY + d)

Rotación circular a la izquierda del contenido de la dirección de memoria especificada por la suma del par IY más el desplazamiento d.

Mnemónico: RLC **Operandos:** (IY + d)

Formato binario:

1	1	1	1	1	1	0	1
1	1	0	0	1	0	1	1
d	d	d	d	d	d	d	d
0	0	0	0	0	1	1	0

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

Tabla indicadores:

S	a 1 si es el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 7

RLA

Rotación a la izquierda del acumulador y el Carry.

Mnemónico: RLA

Operandos: no tiene

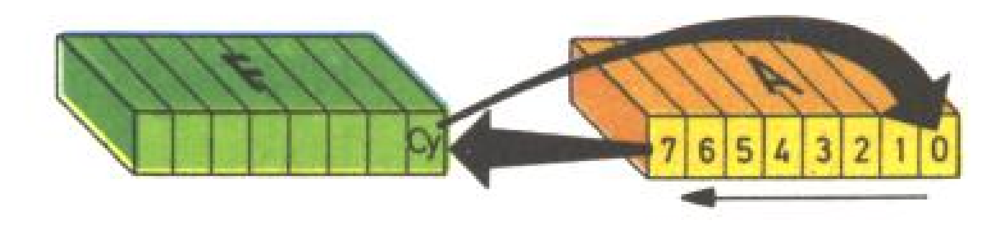
Formato binario:

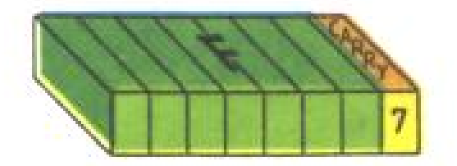
0 0 0 1 0 1 1 1

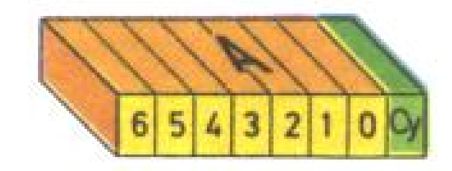
Ciclos: 1
Estados: 4

Indicadores:

S no afectado
Z no afectado
H a 0
P/V no afectado
N a 0
C anterior bit 7







Instr.	Hex.	Dec.
RLA	17	23
RL A	CB,17	203,23
RL B	CB,10	203,16
RL C	CB,11	203,17
RL D	CB,12	203,18
RL E	CB,13	203,19
RLH	CB,14	203,20
RL L	CB,15	203,21
RL (HL)	CB,16	203,22
RL (IX + d)	DD,CB,d,16	221,203,d,22
RL (IY + d)	FD,CB,d,16	253,203,d,22

RL r

Rotación a la izquierda de un registro y el Carry.

Mnemónico: RL

Operandos: r

Formato binario:

1 1 0 0 1 0 1 1
0 0 0 1 0 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla

RL (HL)

Rotación a la izquierda del contenido de la dirección de memoria especificada por el par HL, y el Carry.

Mnemónico: RL **Operandos:** (HL)

Formato binario:

11001011



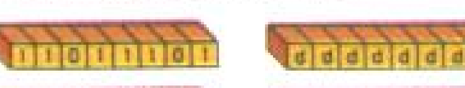
Ciclos: 4
Estados: 15 (4,4,4,3)
Indicadores: ver tabla

RL (IX + d)

Rotación a la izquierda del contenido de la dirección de memoria especificada por la suma del par IX más el desplazamiento d, y el Carry.

Mnemónico: RL **Operandos:** (IX + d)

Formato binario:



11001011



Ciclos: 6
Estados: 23 (4,4,3,5,4,3)
Indicadores: ver tabla

RL (IY + d)

Rotación a la izquierda del contenido de la dirección de memoria especificada por la suma del par IY más el desplazamiento d, y el Carry.

Mnemónico: RL **Operandos:** (IY + d)

Formato binario:

11111101
11001011
dddddddd
00010110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)
Indicadores: ver tabla

Tabla indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 7

RRCA

Rotación circular a la derecha del acumulador. El bit 0 además de pasar al 7 es copiado en el Carry.

Mnemónico: RRCA **Operandos:** no tiene

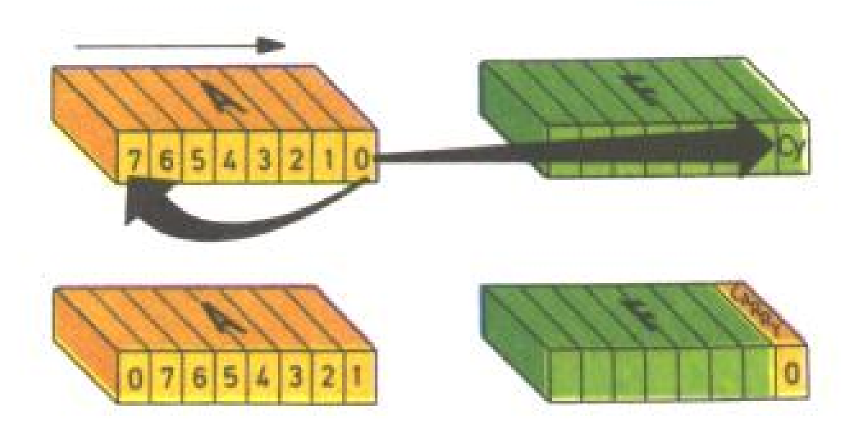
Formato binario:



Ciclos: 1
Estados: 4

Indicadores:

S no afectado
Z no afectado
H a 0
P/V no afectado
N a 0
C anterior bit 0



Instr.	Hex.	Dec.
RRCA	0F	15
RRC A	CB,0F	203,15
RRC B	CB,08	203,8
RRC C	CB,09	203,9
RRC D	CB,0A	203,10
RRC E	CB,0B	203,11
RRC H	CB,0C	203,12
RRC L	CB,0D	203,13
RRC (HL)	CB,0e	203,14
RRC (IX + d)	DD,CB,d,0E	221,203,d,14
RRC (IY + d)	FD,CB,d,0E	253,203,d,14

RRC r

Rotación circular a la derecha de un registro.

Mnemónico: RRC **Operandos:** r

Formato binario:

1 1 0 0 1 0 1 1
0 0 0 0 1 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla

RRC (HL)

Rotación circular a la derecha del contenido de la dirección de memoria especificada por el par HL.

Mnemónico: RRC **Operandos:** (HL)

Formato binario:

1 1 0 0 1 0 1 1

0 0 0 0 1 1 1 0

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla

RRC (IX + d)

Rotación circular a la derecha del contenido de la dirección de memoria especificada por la suma del par IX más el desplazamiento d.

Mnemónico: RRC **Operandos:** (IX + d)

Formato binario:

1 1 0 1 1 1 0 1 d d d d d d d d

1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

RRC (IY + d)

Rotación circular a la derecha del contenido de la dirección de memoria especificada por la suma del par IY más el desplazamiento d.

Mnemónico: RRC **Operandos:** (IY + d)

Formato binario:

1 1 1 1 1 1 0 1

1 1 0 0 1 0 1 1



0 0 0 0 1 1 1 0

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

Tabla indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 0

RRA

Rotación a la derecha del acumulador y el Carry.

Mnemónico: RRA

Operandos: no tiene

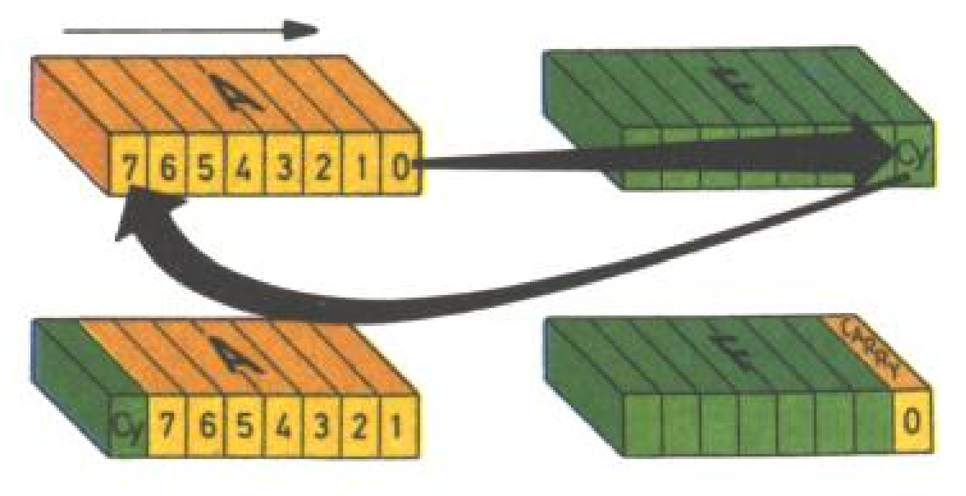
Formato binario:

0 0 0 1 1 1 1 1

Ciclos: 1
Estados: 4

Indicadores:

S no afectado
Z no afectado
H a 0
P / V no afectado
N a 0
C anterior bit 0



Instr.	Hex.	Dec.
RRA	1F	31
RR A	CB,1F	203,31
RR B	CB,18	203,24
RR C	CB,19	203,25
RR D	CB,1A	203,26
RR E	CB,1B	203,27
RR H	CB,1C	203,28
RR L	CB,1D	203,29
RR (HL)	CB,1E	203,30
RR (IX + d)	DD,CB,d,1E	221,203,d,30
RR (IY + d)	FD,CB,d,1E	253,203,d,30

RR r

Rotación a la derecha de un registro y el Carry.

Mnemónico: RR

Operandos: r

Formato binario:

1 1 0 0 1 0 1 1

0 0 0 1 1 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla

RR (HL)

Rotación a la derecha del contenido de la dirección de memoria especificada por el par HL, y el Carry.

Mnemónico: RR **Operandos:** (HL)

Formato binario:

11001011
00011110

Ciclos: 4
Estados: 15 (4,4,4,3)
Indicadores: ver tabla

RR (IX + d)

Rotación a la derecha del contenido de la dirección de memoria especificada por la suma del IX más el desplazamiento d, y el Carry.

Mnemónico: RR **Operandos:** (IX + d)

Formato binario:

11011101
11001011
dddddddd
00011110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)
Indicadores: ver tabla

RR (IY + d)

Rotación a la derecha del contenido de la dirección de memoria especificada porla suma del par IY más el desplazamiento d, y el Carry.

Mnemónico: RR **Operandos:** (IY + d)

Formato binario:

11111101
11001011
dddddddd
00011110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)
Indicadores: ver tabla

Tabla indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterio bit 0

SLA r

Desplazamiento aritmético a la izquierda de un registro.

Mnemónico: SLA **Operandos:** r

Formato binario:

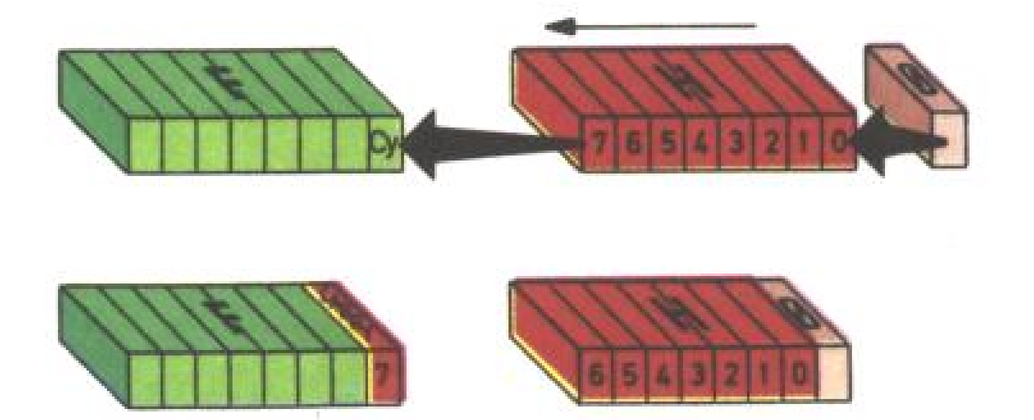
1 1 0 0 1 0 1 1

0 0 1 0 0 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla

Instr.	Hex.	Dec.
SLA A	CB,27	203,39
SLA B	CB,20	203,32
SLA C	CB,21	203,33
SLA D	CB,22	203,34
SLA E	CB,23	203,35
SLA H	CB,24	203,36
SLA L	CB,25	203,37
SLA (HL)	CB,26	203,38
SLA (IX + d)	DD,CB,d,26	221,203,d,38
SLA (IY + d)	FD,CB,d,26	253,203,d,38



Utilización:

Cuando las instrucciones tipo SLA efectúan el desplazamiento, sitúan en el bit 0 un 0 y el bit 7 pasa al carry. Por ello produce una multiplicación por 2.

Si el número que queremos multiplicar por 2 ocupa más de un Byte ha de utilizarse SLA para el Byte menos significativo y RL para los restantes.

SLA (HL)

Desplazamiento aritmético a la izquierda del contenido de la dirección de memoria especificada por el par de registros HL.

Mnemónico: SLA **Operandos:** (HL)

Formato binario:

11001011
00100110

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla

SLA (IX + d)

Desplamiento aritmético a la izquierda del contenido de la dirección de memoria especificada por la suma del par IX y el desplazamiento d.

Mnemónico: SLA **Operandos:** (IX + d)

Formato binario:

11011101
11001011
dddddddd
00100110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

SLA (IY + d)

Desplazamiento aritmético a la izquierda del contenido de la dirección de memoria especificada por la suma del par IY y el desplazamiento d.

Mnemónico:SLA **Operandos:** (IY + d)

Formato binario:

11111101
11001011
dddddddd
00100110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

Tabla indicadores:

S	a 1 el resultado es negativo
Z	a 1 el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 7

SRA r

Desplazamiento aritmético a la derecha de un registro.

Mnemónico: SRA

Operandos: r

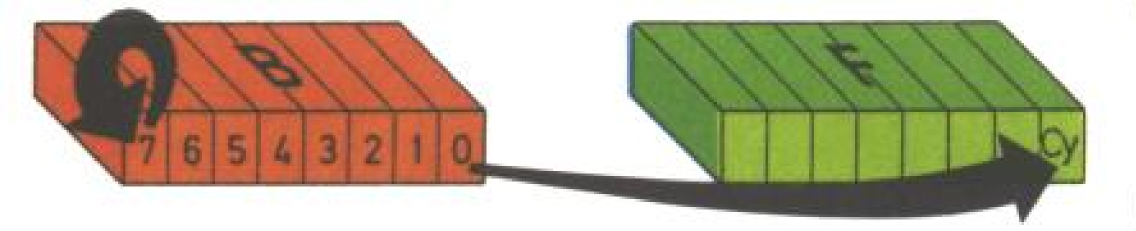
Formato binario:

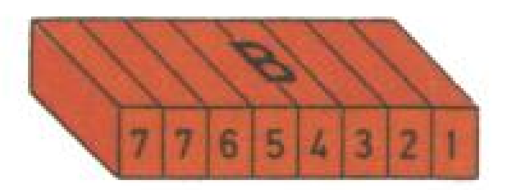
1 1 0 0 1 0 1 1

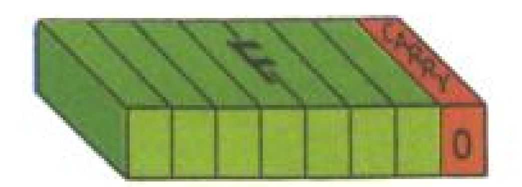
0 0 1 0 1 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tabla







Instr.	Hex.	Dec.
SRA A	CB,2F	203,47
SRA B	CB,28	203,40
SRA C	CB,29	203,41
SRA D	CB,2A	203,42
SRA E	CB,2B	203,43
SRA H	CB,2C	203,44
SRA L	CB,2D	203,45
SRA (HL)	CB,2E	203,46
SRA (IX + d)	DD,CB,d,2E	221,203,d,46
SRA (IY + d)	FD,CB,d,2e	253,203,d,46

Utilización:

Cuando las instrucciones tipo SRA efectúan el desplazamiento, pasan bit 0 al carry y el bit 7 queda como estaba además de ser copiado en el bit 6. Por ello produce una división entre 2 de un número en complemento a 2.

Si el número que queremos dividir entre 2 ocupa más de un Byte ha de utilizarse SRA para el Byte más significativo y RR para los restantes.

SRA (HL)

Desplazamiento aritmético a la derecha del contenido de la dirección de memoria especificada por el par de registros HL.

Mnemónico: SRA **Operandos:** (HL)

Formato binario:

11001011
00101110

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla

SRA (IX + d)

Desplazamiento aritmético a la derecha del contenido de la dirección de memoria especificada por la suma del par IX y el desplazamiento d.

Mnemónico: SRA **Operandos:** (IX + d)

Formato binario:

11011101 dddddddd
11001011 00101110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

SRA (IY + d)

Desplazamiento aritmético a la derecha del contenido de la dirección de memoria especificada por la suma del par IY y el desplazamiento d.

Mnemónico: SRA **Operandos:** (IY + d)

Formato binario:

11111101 dddddddd
11001011 00101110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)
Indicadores: ver tabla

Tabla indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/V	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 0

SRL r

Desplazamiento lógico a la derecha de un registro.

Mnemónico: SRL **Operandos:** r

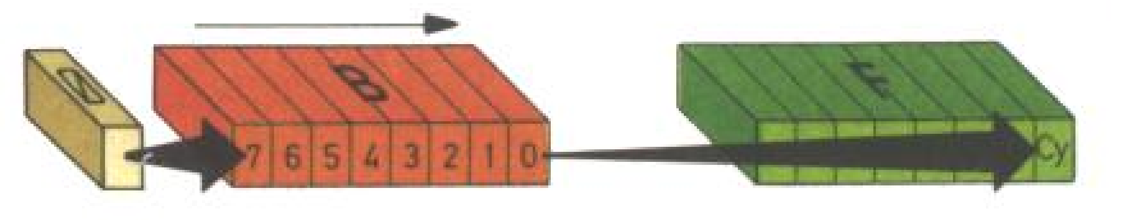
Formato binario:

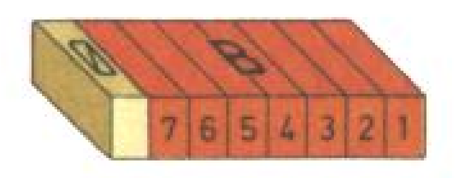
1 1 0 0 1 0 1 1

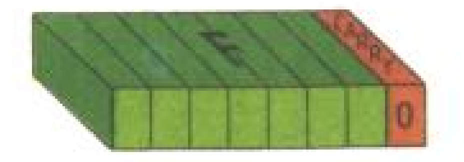
0 0 1 1 1 r r r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ver tablas







Instr.	Hext.	Dec.
SRL A	CB,3F	203,63
SRL B	CB,38	203,56
SRL C	CB,39	203,57
SRL D	CB,3A	203,58
SRL E	CB,3B	203,59
SRL H	CB,3C	203,60
SRL L	CB,3D	203,61
SRL (HL)	CB,3E	203,62
SRL (IX + d)	DD,CB,d,3E	221,203,d,62
SRL (IY + d)	FD,CB,d,3E	253,203,d,62

Utilización:

Cuando las instrucciones tipo SRL efectúan el desplazamiento, sitúan en el bit 7 un 0 y el bit 0 pasa al Carry. Por ello produce una división entre 2 de un número positivo de 8 bits.

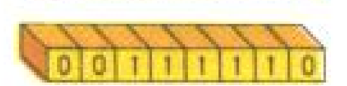
Si el número que queremos dividir entre 2 ocupa más de un Byte ha de utilizarse SRL para el Byte más significativo y RR para los restantes.

SRL (HL)

Desplazamiento lógico a la derecha del contenido de la dirección de memoria especificada por el par de registros HL.

Mnemónico: SRL **Operandos:** (HL)

Formato binario:



Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ver tabla

SRL (IX + d)

Desplazamiento lógico a la derecha del contenido de la dirección de memoria especificada por la suma del par IX y el desplazamiento D.

Mnemónico: SRL **Operandos:** (IX + d)

Formato binario:

11011101
11001011
dddddddd
00111110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

SRL (IY + d)

Desplazamiento lógico a la derecha del contenido de la dirección de memoria especificada por la suma del par IY y el desplazamiento d.

Mnemónico: SRL **Operandos:** (IY + d)

Formato binario:

11111101
11001011
dddddddd
00111110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ver tabla

Tabla indicadores:

S	a 1 si el resultado es negativo
Z	a 1 si el resultado es cero
H	a 0
P/H	a 1 si hay paridad (par)
N	a 0
C	como el anterior bit 0

RLD RRD

RLD

Rotación decimal a la izquierda: Los cuatro bits bajos de la dirección de memoria especificada por el par HL son copiados en la parte alta de la misma, los cuatro bits altos son copiados en la parte baja del registro A y la parte baja del acumulador es copiada en la parte baja de aquella dirección.

Mnemónico: RLD **Operandos:** no tiene

Formato binario:



Ciclos: 5
Estados: 18 (4,4,3,4,3)

Indicadores: ver tabla

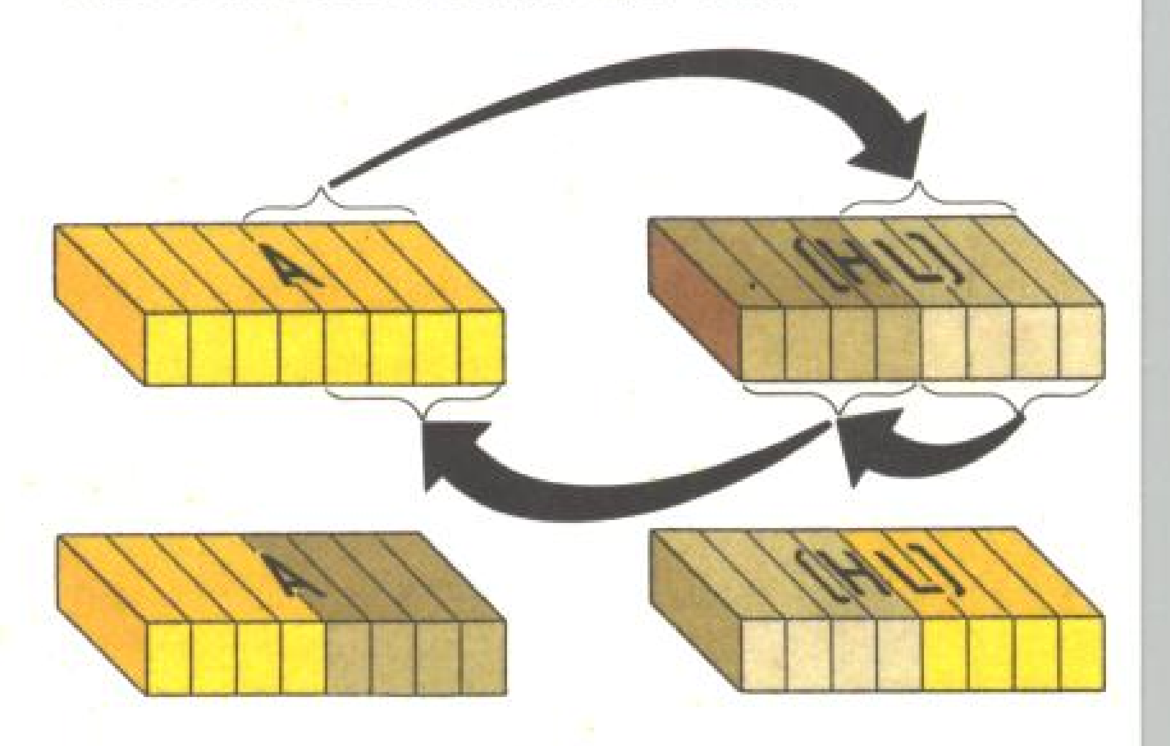
Ejemplo:

Si el registro A contiene 3AH, el par HL 2000H y la dirección de memoria 2000H contine C1H, después de la instrucción

RLD

Instr.	Hex.	Dec.
RLD	ED,6F	237,111
RRD	ED,67	237,103

el registro A contendrá 3CH y la dirección de memoria 2000H contendrá 1AH.



RRD

Rotación decimal a la derecha: Los cuatro bits altos de la dirección de memoria especificada por el par HL son copiados en la parte baja de la misma, los cuatro bits bajos son copiados en la parte baja del registro A y la parte baja del acumulador es copiada en la parte alta de aquella dirección.

Mnemónico: RRD **Operandos:** no tiene

Formato binario:

1 1 1 0 1 1 0 1

0 1 1 0 0 1 1 1

Ciclos: 5
Estados: 18 (4,4,3,4,3)

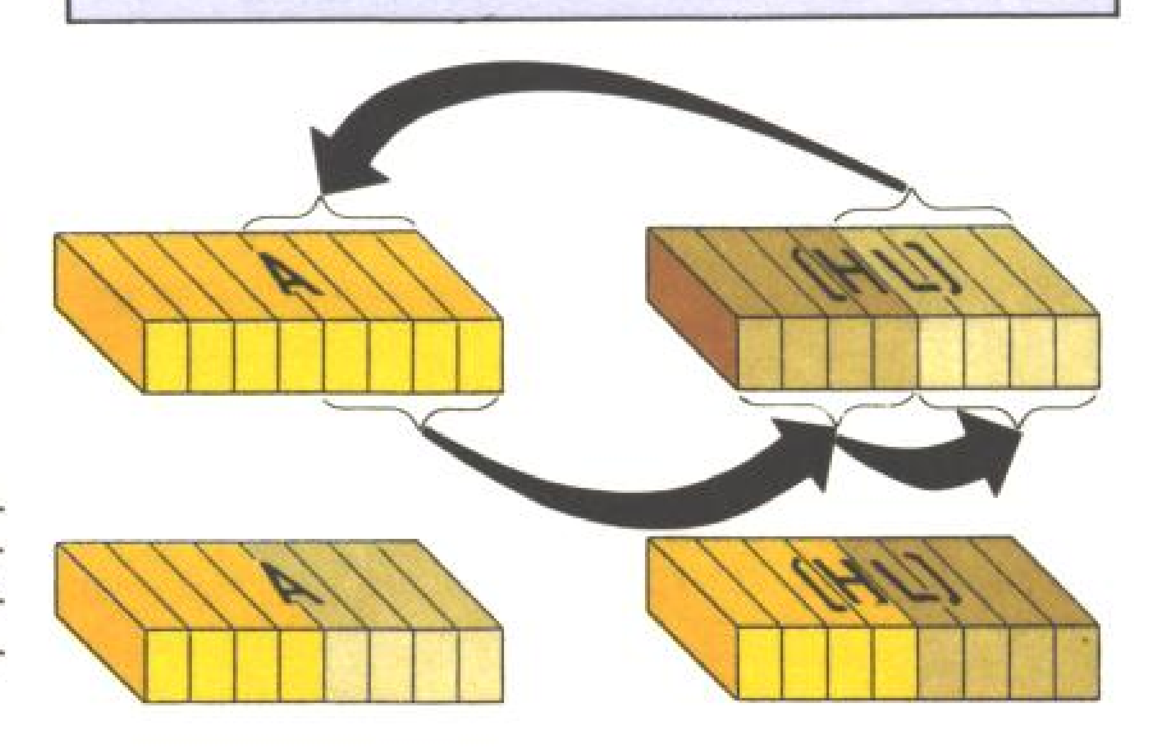
Indicadores: ver tabla

Ejemplo:

Si el registro A contiene D5H, el par HL FFFFH y la dirección de memoria FFFFH contiene C1H, después de la instrucción RRD el registro A contendrá D1H y la dirección de memoria FFFFH contendrá 5CH.

Tabla indicadores:

S	a 1 si el acumulador es negativo
Z	a 1 si el acumulador resulta ser cero
H	a 0
P/V	a 1 si hay paridad en el acumulador
N	a 0
C	no afectado



BIT b,r

Comprobación del estado de un determinado bit de un registro. Después de la ejecución de esta instrucción, el flag Z del registro de indicadores F contendrá el complemento del bit en concreto del registro determinado por la instrucción.

Los operandos b y r son especificados implícitamente en un solo byte del código objeto, por lo que no es posible un direccionamiento indirecto de bit.

Mnemónico: BIT **Operandos:** b,r

Formato binario:

11001011

01bbbrrr

Ciclos: 2
Estados: 8 (4,4)

Indicadores:

S	desconocido	P/V	desconocido
Z	a 1 si el bit especificado es 0	N	es 0
H	a 1	C	no afectado

Instr.	Hex.	Dec.
BIT 0,B	CB,40	203,64
BIT 0,C	CB,41	203,65
BIT 0,D	CB,42	203,66
BIT 0,E	CB,43	203,67
BIT 0,H	CB,44	203,68
BIT 0,L	CB,45	203,69
BIT 0,A	CB,47	203,71
BIT 1,B	CB,48	203,72
BIT 1,C	CB,49	203,73
BIT 1,D	CB,4A	203,74
BIT 1,E	CB,4B	203,75
BIT 1,H	CB,4C	203,76
BIT 1,L	CB,4D	203,77
BIT 1,A	CB,4F	203,79
BIT 2,B	CB,50	203,80
BIT 2,C	CB,51	203,81
BIT 2,D	CB,52	203,82
BIT 2,E	CB,53	203,83
BIT 2,H	CB,54	203,84
BIT 2,L	CB,55	203,85
BIT 2,A	CB,57	203,87
BIT 3,B	CB,58	203,88
BIT 3,C	CB,59	203,89
BIT 3,D	CB,5A	203,90
BIT 3,E	CB,5B	203,91
BIT 3,H	CB,5C	203,92
BIT 3,L	CB,5D	203,93
BIT 3,A	CB,5F	203,95

Instr.	Hex.	Dec.
BIT 4,B	CB,60	203,96
BIT 4,C	CB,61	203,97
BIT 4,D	CB,62	203,98
BIT 4,E	CB,63	203,99
BIT 4,H	CB,64	203,100
BIT 4,L	CB,65	203,101
BIT 4,A	CB,67	203,103
BIT 5,B	CB,68	203,104
BIT 5,C	CB,69	203,105
BIT 5,D	CB,6A	203,106
BIT 5,E	CB,6B	203,107
BIT 5,H	CB,6C	203,108
BIT 5,L	CB,6D	203,109
BIT 5,A	CB,6F	203,111
BIT 6,B	CB,70	203,112
BIT 6,C	CB,71	203,113
BIT 6,D	CB,72	203,114
BIT 6,E	CB,73	203,115
BIT 6,H	CB,74	203,116
BIT 6,L	CB,75	203,117
BIT 6,A	CB,77	203,119
BIT 7,B	CB,78	203,120
BIT 7,C	CB,79	203,121
BIT 7,D	CB,7A	203,122
BIT 7,E	CB,7B	203,123
BIT 7,H	CB,7C	203,124
BIT 7,L	CB,7D	203,125
BIT 7,A	CB,7F	203,127

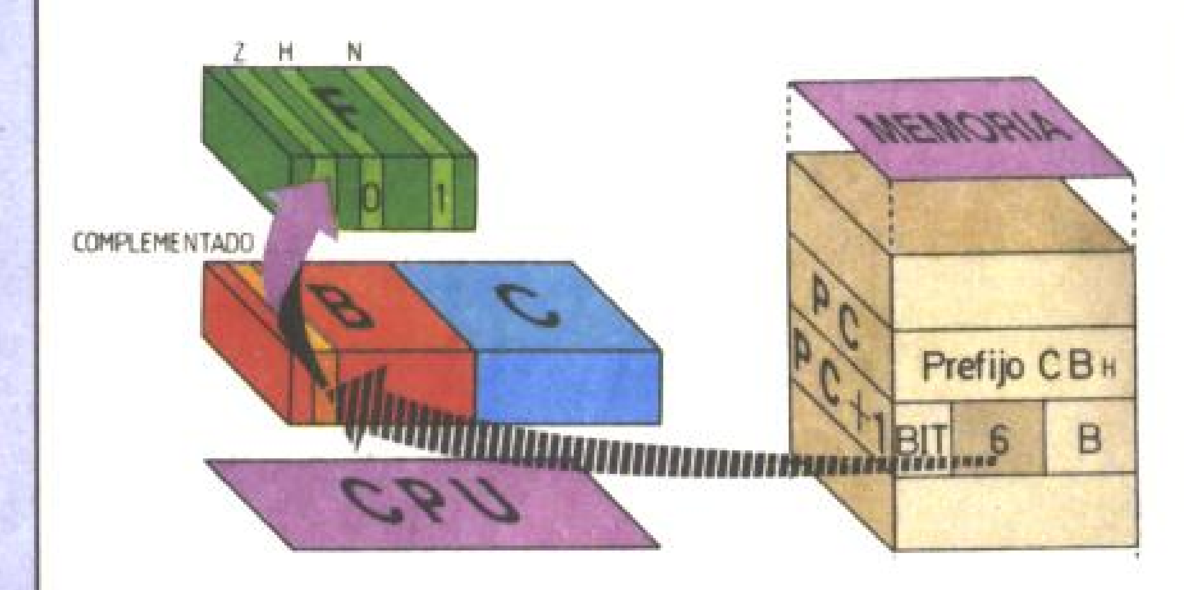
Ejemplo:

Si el registro B contiene 3DH (00111101b) la secuencia de instrucciones:

```
BIT     6,B
CALL    Z,RUT
```

pondrá a 1 el indicador Z del registro F, porque el bit 6 del registro B es 0.

Posteriormente, debido a esto, la rutina «RUT» será ejecutada.



BIT b, (HL)

El flag Z del registro de indicadores F toma el valor del complemento de un bit concreto en la posición de memoria señalada por el par de registros HL.

Mnemónico: BIT

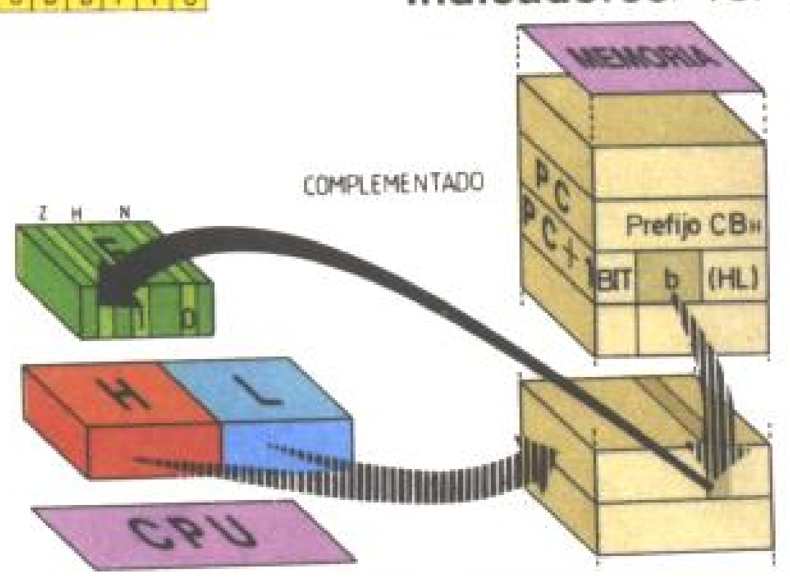
Operandos: b,(HL)

Formato binario:

1	1	0	0	1	0	1	1
0	1	b	b	b	1	1	0

Ciclos: 3
Estados: 12 (4,4,4)

Indicadores: ver tabla



Instr.	Hex.	Dec.
BIT 0 (HL)	CB,46	203,70
BIT 1 (HL)	CB,4E	203,78
BIT 2 (HL)	CB,56	203,86
BIT 3 (HL)	CB,5E	203,94
BIT 4 (HL)	CB,66	203,102
BIT 5 (HL)	CB,6E	203,110
BIT 6 (HL)	CB,76	203,118
BIT 7 (HL)	CB,7E	203,126
BIT 0 (IX + d)	DD,CB,d,46	221,203,d,70
BIT 1 (IX + d)	DD,CB,d,4E	221,203,d,78
BIT 2 (IX + d)	DD,CB,d,56	221,203,d,86
BIT 3 (IX + d)	DD,CB,d,5E	221,203,d,94
BIT 4 (IX + d)	DD,CB,d,66	221,203,d,102
BIT 5 (IX + d)	DD,CB,d,6E	221,203,d,110
BIT 6 (IX + d)	DD,CB,d,76	221,203,d,118
BIT 7 (IX + d)	DD,CB,d,7E	221,203,d,126
BIT 0 (IY + d)	FD,CB,d,46	253,203,d,70
BIT 1 (IY + d)	FD,CB,d,4E	253,203,d,78
BIT 2 (IY + d)	FD,CB,d,56	253,203,d,86
BIT 3 (IY + d)	FD,CB,d,5E	253,203,d,94
BIT 4 (IY + d)	FD,CB,d,66	253,203,d,102
BIT 5 (IY + d)	FD,CB,d,6E	253,203,d,110
BIT 6 (IY + d)	FD,CB,d,76	253,203,d,118
BIT 7 (IY + d)	FD,CB,d,7E	253,203,d,126

BIT b,(IX + d)

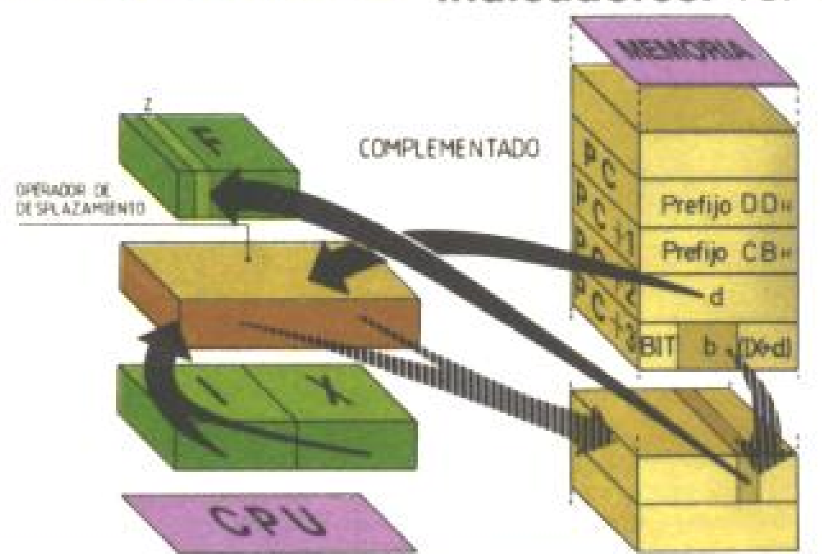
El flag Z del registro de indicadores F toma el valor del complemento de un bit concreto en la posición de memoria especificada por la suma del contenido del par IX y el desplazamiento d.

Mnemónico: BIT **Operandos:** b,(IX + d)

Formato binario:

11011101 dddddddd
11001011 01bbb110

Ciclos: 3
Estados: 20 (4,4,3,5,4)
Indicadores: ver tabla



BIT b, (IY + d)

El flag Z del registro de indicadores F toma el valor del complemento de un bit concreto en la posición de memoria especificada por la suma del contenido del par IY y el desplazamiento d.

Mnemónico: BIT **Operandos:** b, (IY + d)

Formato binario:

11111101 dddddddd
11001011 01bbb110

Ciclos: 3
Estados: 20 (4,4,3,5,4)
Indicadores: ver tabla

Tabla de indicadores:

S	desconocido
Z	a 1 si el bit especificado es 0
H	a 1
P/V	desconocido
N	a 0
C	no afectado

SET b,r

Asignación del valor 1 a un determinado bit de un registro. Después de la ejecución de esta instrucción el bit en concreto del registro indicado por la instrucción contendrá un 1 mientras que los restantes continuarán con su anterior valor.

Los operandos b y r son especificados implícitamente en un solo byte del código objeto, por lo que no es posible un direccionamiento indirecto de bit.

Mnemónico: SET

Operandos: b,r

Formato binario:

1	1	0	0	1	0	1	1
1	0	b	b	b	r	r	r

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ninguno

Instr.	Hex.	Dec.
SET 0,B	CB,C0	203,192
SET 0,C	CB,C1	203,193
SET 0,D	CB,C2	203,194
SET 0,E	CB,C3	203,195
SET 0,H	CB,C4	203,196
SET 0,L	CB,C5	203,197
SET 0,A	CB,C7	203,199
SET 1,B	CB,C8	203,200
SET 1,C	CB,C9	203,201
SET 1,D	CB,CA	203,202
SET 1,E	CB,CB	203,203
SET 1,H	CB,CC	203,204
SET 1,L	CB,CD	203,205
SET 1,A	CB,CF	203,207
SET 2,B	CB,D0	203,208
SET 2,C	CB,D1	203,209
SET 2,D	CB,D2	203,210
SET 2,E	CB,D3	203,211
SET 2,H	CB,D4	203,212
SET 2,L	CB,D5	203,213
SET 2,A	CB,D7	203,215
SET 3,B	CB,D8	203,216
SET 3,C	CB,D9	203,217
SET 3,D	CB,DA	203,218
SET 3,E	CB,DB	203,219
SET 3,H	CB,DC	203,220
SET 3,L	CB,DD	203,221
SET 3,A	CB,DF	203,223

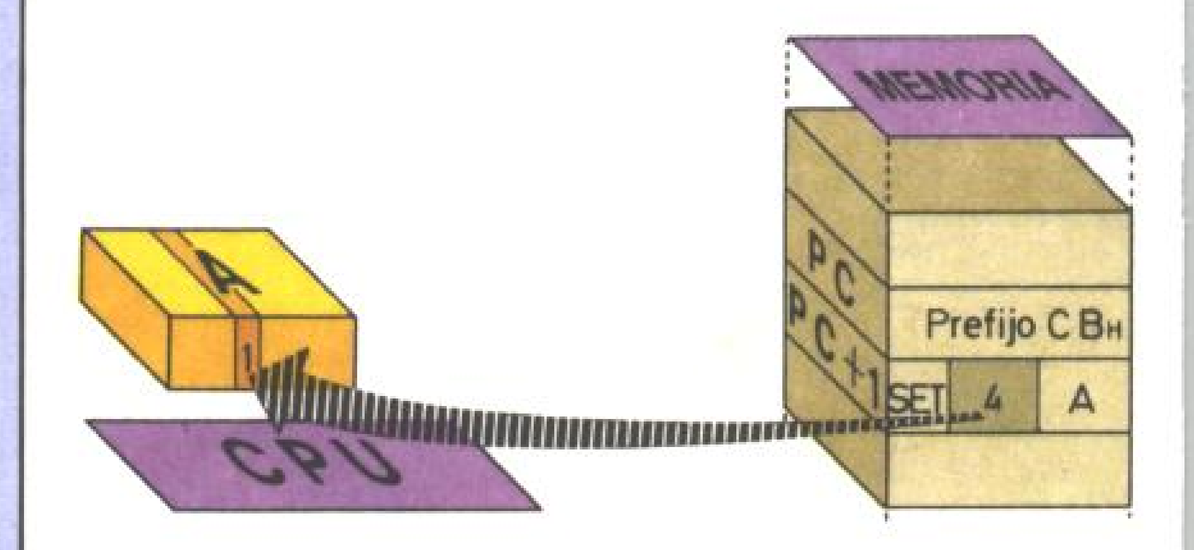
Instr.	Hex.	Dec.
SET 4,B	CB,E0	203,224
SET 4,C	CB,E1	203,225
SET 4,D	CB,E2	203,226
SET 4,E	CB,E3	203,227
SET 4,H	CB,E4	203,228
SET 4,L	CB,E5	203,229
SET 4,A	CB,E7	203,231
SET 5,B	CB,E8	203,232
SET 5,C	CB,E9	203,233
SET 5,D	CB,EA	203,234
SET 5,E	CB,EB	203,235
SET 5,H	CB,EC	203,236
SET 5,L	CB,ED	203,237
SET 5,A	CB,EF	203,239
SET 6,B	CB,F0	203,240
SET 6,C	CB,F1	203,241
SET 6,D	CB,F2	203,242
SET 6,E	CB,F3	203,243
SET 6,H	CB,F4	203,244
SET 6,L	CB,F5	203,245
SET 6,A	CB,F7	203,247
SET 7,B	CB,F8	203,248
SET 7,C	CB,F9	203,249
SET 7,D	CB,FA	203,250
SET 7,E	CB,FB	203,251
SET 7,H	CB,FC	203,252
SET 7,L	CB,FD	203,253
SET 7,A	CB,FF	203,255

Ejemplo:

Si el registro A contiene 8FH (10001111b), después de la instrucción:

SET 4,A

habrá un 1 en el bit 4 del acumulador quedando los demás como estaban. El registro A resultará con el valor 9FH (10011111b).



SET b,(HL)

Asigna el valor 1 a un bit en concreto de la posición de memoria señalada por el par de registros HL.

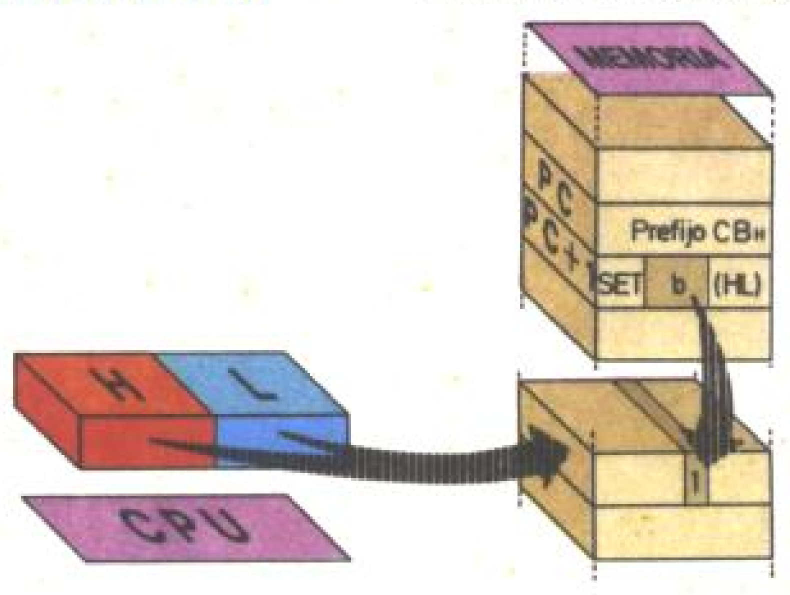
Mnemónico: SET **Operandos:** B,(HL)

Formato binario:

1	1	0	0	1	0	1	1
1	0	b	b	b	1	1	0

Ciclos: 4
Estados: 15 (4,4,4,3)

Indicadores: ninguno



Instr.	Hex.	Dec.
SET 0 (HL)	CB,C6	203,198
SET 1 (HL)	CB,CE	203,206
SET 2 (HL)	CB,D6	203,214
SET 3 (HL)	CB,DE	203,222
SET 4 (HL)	CB,E6	203,230
SET 5 (HL)	CB,EE	203,238
SET 6 (HL)	CB,F6	203,246
SET 7 (HL)	CB,FE	203,254
SET 0 (IX + d)	DD,CB,d,C6	221,203,d,198
SET 1 (IX + d)	DD,CB,d,CE	221,203,d,206
SET 2 (IX + d)	DD,CB,d,D6	221,203,d,214
SET 3 (IX + d)	DD,CB,d,DE	221,203,d,222
SET 4 (IX + d)	DD,CB,d,E6	221,203,d,230
SET 5 (IX + d)	DD,CB,d,EE	221,203,d,238
SET 6 (IX + d)	DD,CB,d,F6	221,203,d,246
SET 7 (IX + d)	DD,CB,d,FE	221,203,d,254
SET 0 (IY + d)	FD,CB,d,C6	253,203,d,198
SET 1 (IY + d)	FD,CB,d,CE	253,203,d,206
SET 2 (IY + d)	FD,CB,d,D6	253,203,d,214
SET 3 (IY + d)	FD,CB,d,DE	253,203,d,222
SET 4 (IY + d)	FD,CB,d,E6	253,203,d,230
SET 5 (IY + d)	FD,CB,d,EE	253,203,d,238
SET 6 (IY + d)	FD,CB,d,F6	253,203,d,246
SET 7 (IY + d)	FD,CB,d,FE	253,203,d,254

SET b,(IX + d)

Asigna el valor 1 a un bit en concreto de la posición de memoria especificada por la suma del contenido del par IX y el desplazamiento d.

Mnemónico: SET

Operandos: b,(IX + d)

Formato binario:

1 1 0 1 1 1 0 1

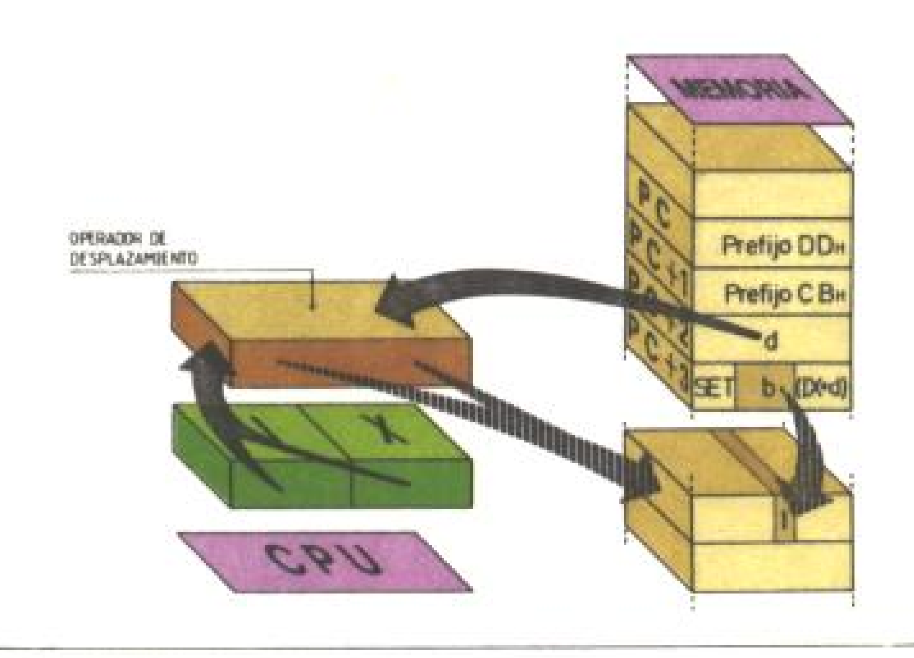
1 1 0 0 1 0 1 1

d d d d d d d d

1 0 b b b 1 1 0

Ciclos: 5
Estados: 20 (4,4,3,5,4)

Indicadores: ninguno



SET b,(IY + d)

Asigna el valor 1 a un bit en concreto de la posición de memoria especificada por la suma del contenido del par IY y el desplazamiento d.

Mnemónico: SET

Operandos: b,(IY + d)

Formato binario:

1 1 1 1 1 1 0 1

1 1 0 0 1 0 1 1

d d d d d d d d

1 0 b b b 1 1 0

Ciclos: 5
Estados: 20 (4,4,3,5,4)

Indicadores: ninguno

RES b,r

Asignación del valor 0 a un determinado bit de un registro. Después de la ejecución de esta instrucción el bit en concreto del registro indicado por la instrucción contendrá un 0 mientras que los restantes continuarán con su anterior valor.

Los operandos b y r son especificados implícitamente en un solo byte del código objeto, por lo que no es posible un direccionamiento indirecto de bit.

Mnemónico: RES

Operandos: b,r

Formato binario:

1	1	0	0	1	0	1	1

1	1	b	b	b	r	r	r

Ciclos: 4
Estados: 8 (4,4)

Indicadores: ninguno

Instr.	Hex.	Dec.
RES 0,B	CB,80	203,128
RES 0,C	CB,81	203,129
RES 0,D	CB,82	203,130
RES 0,E	CB,83	203,131
RES 0,H	CB,84	203,132
RES 0,L	CB,85	203,133
RES 0,A	CB,87	203,135
RES 1,B	CB,88	203,136
RES 1,C	CB,89	203,137
RES 1,D	CB,8A	203,138
RES 1,E	CB,8B	203,139
RES 1,H	CB,8C	203,140
RES 1,L	CB,8D	203,141
RES 1,A	CB,8F	203,143
RES 2,B	CB,90	203,144
RES 2,C	CB,91	203,145
RES 2,D	CB,92	203,146
RES 2,E	CB,93	203,147
RES 2,H	CB,94	203,148
RES 2,L	CB,95	203,149
RES 2,A	CB,97	203,151
RES 3,B	CB,98	203,152
RES 3,C	CB,99	203,153
RES 3,D	CB,9A	203,154
RES 3,E	CB,9B	203,155
RES 3,H	CB,9C	203,156
RES 3,L	CB,9D	203,157
RES 3,A	CB,9F	203,159

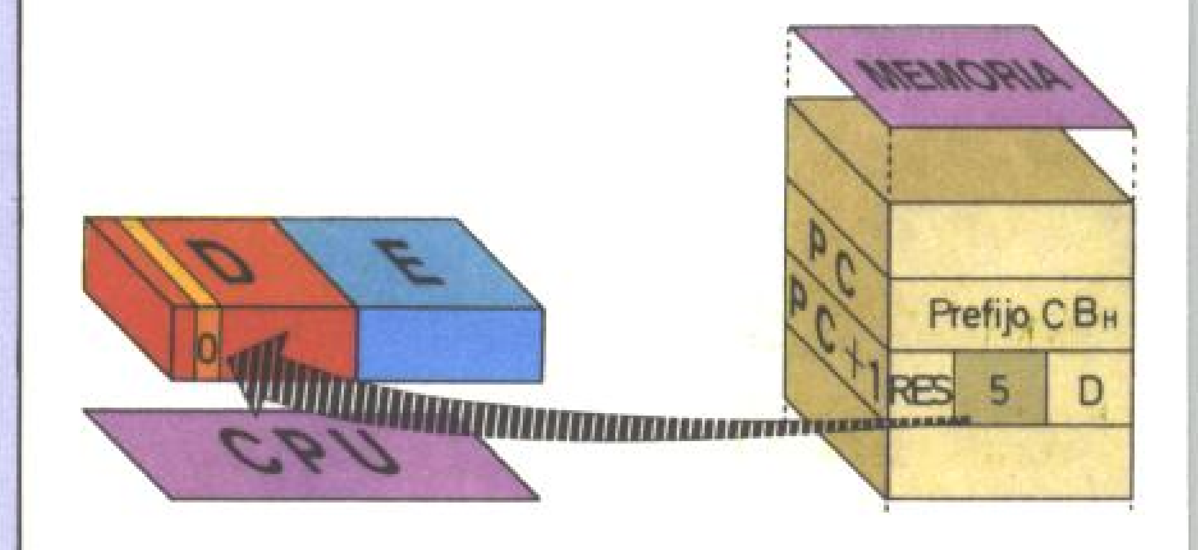
Instr.	Hex.	Dec.
RES 4,B	CB,A0	203,160
RES 4,C	CB,A1	203,161
RES 4,D	CB,A2	203,162
RES 4,E	CB,A3	203,163
RES 4,H	CB,A4	203,164
RES 4,L	CB,A5	203,165
RES 4,A	CB,A7	203,167
RES 5,B	CB,A8	203,168
RES 5,C	CB,A9	203,169
RES 5,D	CB,AA	203,170
RES 5,E	CB,AB	203,171
RES 5,H	CB,AC	203,172
RES 5,L	CB,AD	203,173
RES 5,A	CB,AF	203,175
RES 6,B	CB,B0	203,176
RES 6,C	CB,B1	203,177
RES 6,D	CB,B2	203,178
RES 6,E	CB,B3	203,179
RES 6,H	CB,B4	203,180
RES 6,L	CB,B5	203,181
RES 6,A	CB,B7	203,183
RES 7,B	CB,B8	203,184
RES 7,C	CB,B9	203,185
RES 7,D	CB,BA	203,186
RES 7,E	CB,BB	203,187
RES 7,H	CB,BC	203,188
RES 7,L	CB,BD	203,189
RES 7,A	CB,BF	203,191

Ejemplo:

Si el registro D contiene F6H (11110110b), después de la instrucción:

RES 5,D

habrá un 0 en el bit 5 del registro D quedando los demás como estaban, resultando finalmente con el valor D6H (11010110b).



RES b,(HL)

Asigna el valor 0 a un bit en concreto de la posición de memoria señalada por el par de registros HL.

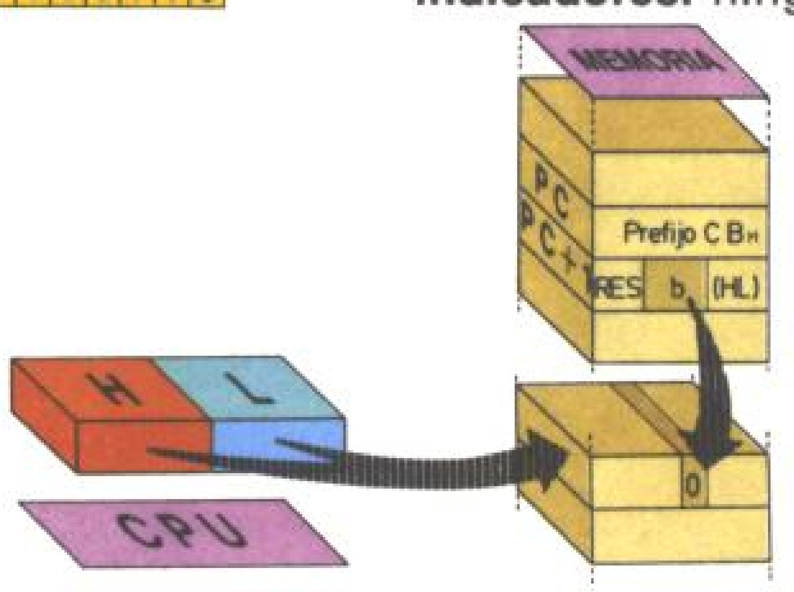
Mnemónico: RES

Operandos: b, (HL)

Formato binario:

Ciclos: 4
Estados: 15 (4,4,4,3)

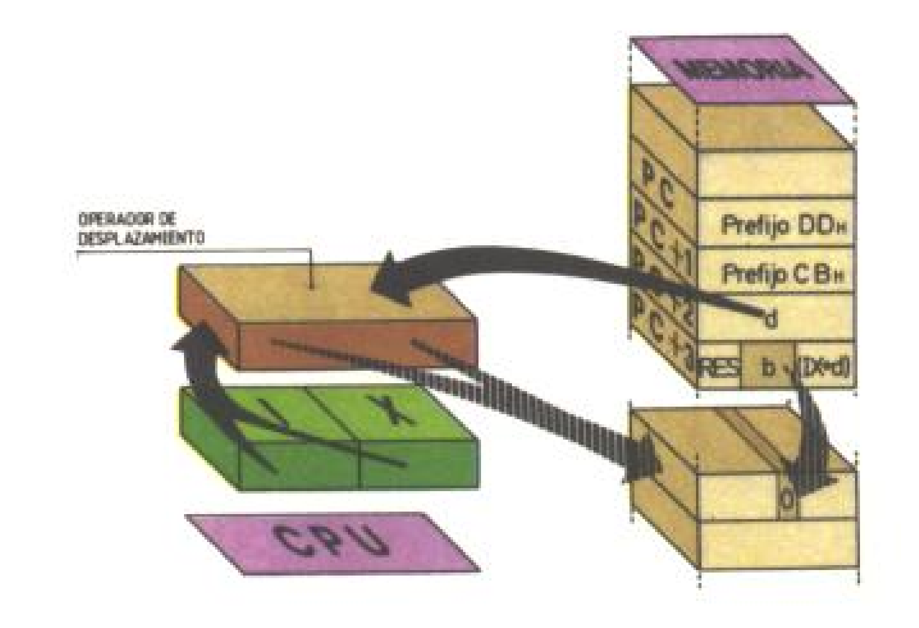
Indicadores: ninguno



Instr.	Hex.	Dec.
RES 0 (HL)	CB,86	203,134
RES 1 (HL)	CB,8E	203,142
RES 2 (HL)	CB,96	203,150
RES 3 (HL)	CB,9E	203,158
RES 4 (HL)	CB,A6	203,166
RES 5 (HL)	CB,AE	203,174
RES 6 (HL)	CB,B6	203,182
RES 7 (HL)	CB,BE	203,190
RES 0 (IX + d)	DD,CB,d,86	221,203,d,134
RES 1 (IX + d)	DD,CB,d,8E	221,203,d,142
RES 2 (IX + d)	DD,CB,d,96	221,203,d,150
RES 3 (IX + d)	DD,CB,d,9E	221,203,d,158
RES 4 (IX + d)	DD,CB,d,A6	221,203,d,166
RES 5 (IX + d)	DD,CB,d,AE	221,203,d,174
RES 6 (IX + d)	DD,CB,d,B6	221,203,d,182
RES 7 (IX + d)	DD,CB,d,BE	221,203,d,190
RES 0 (IY + d)	FD,CB,d,86	253,203,d,134
RES 1 (IY + d)	FD,CB,d,8E	253,203,d,142
RES 2 (IY + d)	FD,CB,d,96	253,203,d,150
RES 3 (IY + d)	FD,CB,d,9E	253,203,d,158
RES 4 (IY + d)	FD,CB,d,A6	253,203,d,166
RES 5 (IY + d)	FD,CB,d,AE	253,203,d,174
RES 6 (IY + d)	FD,CB,d,B6	253,203,d,182
RES 7 (IY + d)	FD,CB,d,BE	253,203,d,190

RES b,(IX + d)

Asigna el valor 0 a un bit en concreto de la posición de memoria especificada por la suma del contenido del par IX y el desplazamiento d.



Mnemónico: RES

Operandos: b,(IX + d)

Formato binario:

11011101
11001011
dddddddd
11bbb110

Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ninguno

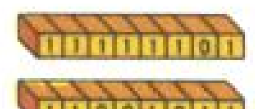
RES b,(IY + d)

Asigna el valor 0 a un bit en concreto de la posición de memoria especificada por la suma del contenido del par IY y el desplazamiento d.

Mnemónico: RES

Operandos: b,(IY + d)

Formato binario:





Ciclos: 6
Estados: 23 (4,4,3,5,4,3)

Indicadores: ninguno

JP nn

El número «nn» de 32 bits es transferido al registro contador de programa PC, saltando a aquella dirección la ejecución del programa.

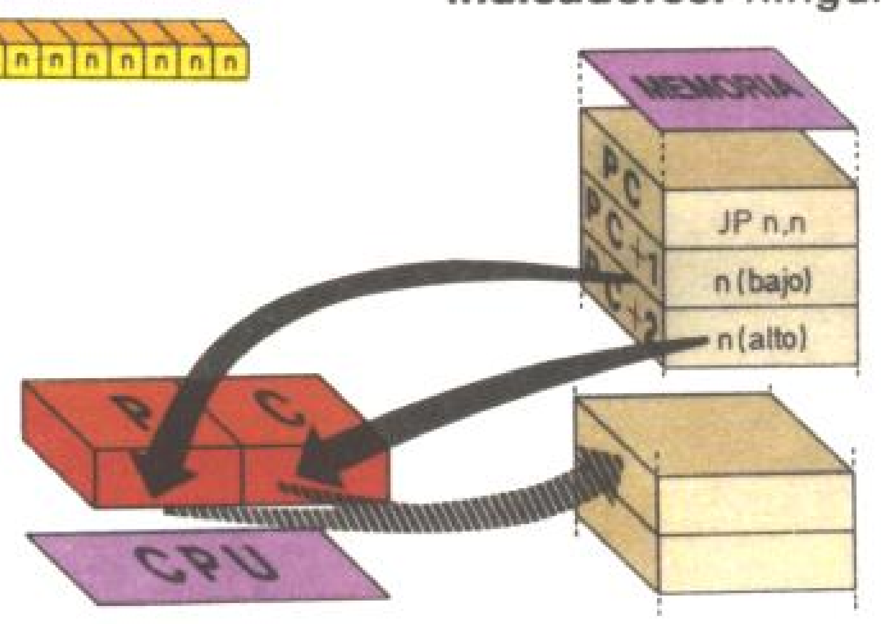
Mnemónico: JP **Operandos:** nn

Formato binario:

1	1	0	0	0	0	1	1
n	n	n	n	n	n	n	n
n	n	n	n	n	n	n	n

Ciclos: 3
Estados: 10 (4,3,3)

Indicadores: ninguno



Instr.	Hex.	Dec.
JP nn	C3,n,n	195,n,n
JP NZ,nn	C2,n,n	194,n,n
JP Z,nn	CA,n,n	202,n,n
JP NC,nn	D2,n,n	210,n,n
JP C,nn	DA,n,n	218,n,n
JP PO,nn	E2,n,n	226,n,n
JP PE,nn	EA,n,n	234,n,n
JP P,nn	F2,n,n	242,n,n
JP M,nn	FA,n,n	250,n,n

Ejemplo:

Después de la instrucción:

JP 23FAH

el registro PC contendrá 23FAH y a continuación no se ejecutará la instrucción siguiente sino la situada en la dirección 23FAH.

JP cc,nn

Si la condición «cc» se cumple, el número «nn» de 32 bits es transferido al registro contador de programa PC, saltando a aquella dirección la ejecución del programa.

Mnemónico: JP **Operandos:** cc,nn

Formato binario:

Ciclos: 3
Estados: 10 (4,3,3)

Indicadores: ninguno



Tabla de Condiciones

cc	Condición		Flag	
000	NZ	no cero	Z	(=0)
001	Z	cero	Z	(=1)
010	NC	no carry	C	(=0)
011	C	carry	C	(=1)
100	PO	paridad impar	P/V	(=0)
101	PE	paridad par	P/V	(=1)
110	P	signo positivo	S	(=0)
111	M	signo negativo	S	(=1)

Ejemplo:

Si el registro E contiene FFH después de la secuencia de instrucciones:

```
INC    E
JP     Z,1A3FH
```

el registro E contendrá 0 y el registro PC contendrá 1A3FH y a continuación se ejecutará la instrucción situada en aquella dirección.

Si el registro E contiene cualquier otro valor no se produce el salto.

JP (HL)

El contenido del par de registros HL es transferido al registro contador de programa PC, saltando a aquella dirección la ejecución del programa.

Instr.	Hex.	Dec.
JP (HL)	E9	233
JP (IX)	DD,E9	221,233
JP (IY)	FD,E9	253,233

Mnemónico: JP **Operandos:** (HL)

Formato binario:

1	1	1	0	1	0	0	1

Ciclos: 1
Estados: 4

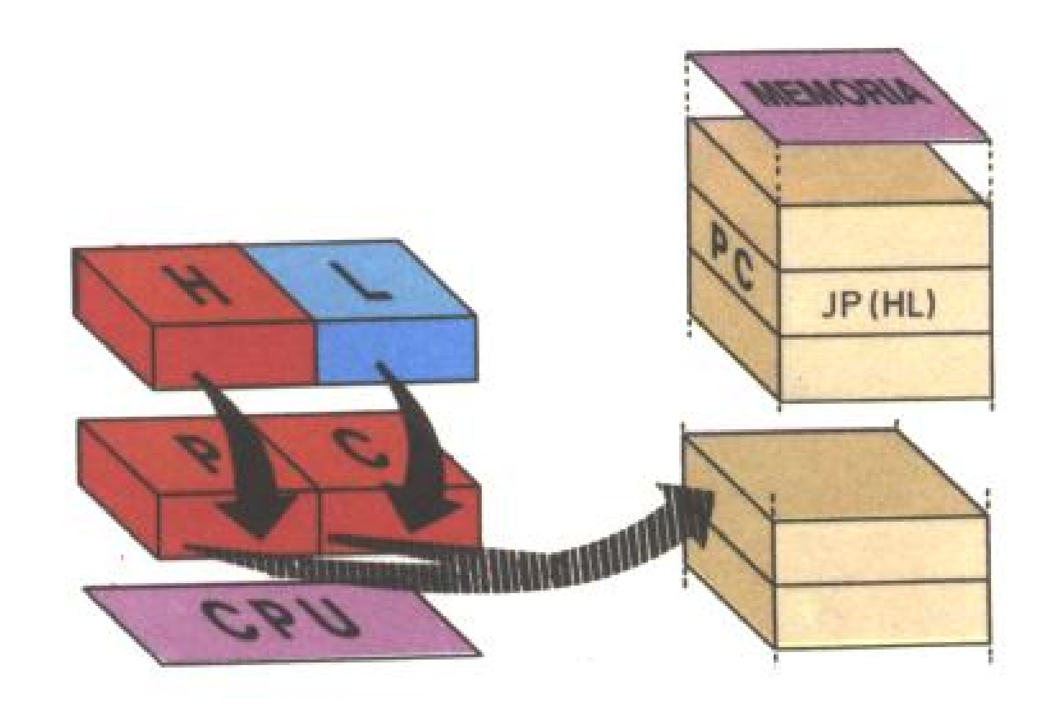
Indicadores: ninguno

Ejemplo:

Si el par de registros HL contiene la dirección 3AF5H, después de ejecutar la instrucción:

JP (HL)

el registro PC contendrá 34F5H. Debido a esto, a continuación no se ejecutará la instrucción siguiente sino la situada en la dirección 3AF5H.

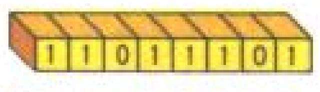


JP (IX)

El contenido del registro índice IX es transferido al registro contador de programa PC, saltando a aquella dirección la ejecución del programa.

Mnemónico: JP **Operandos:** (IX)

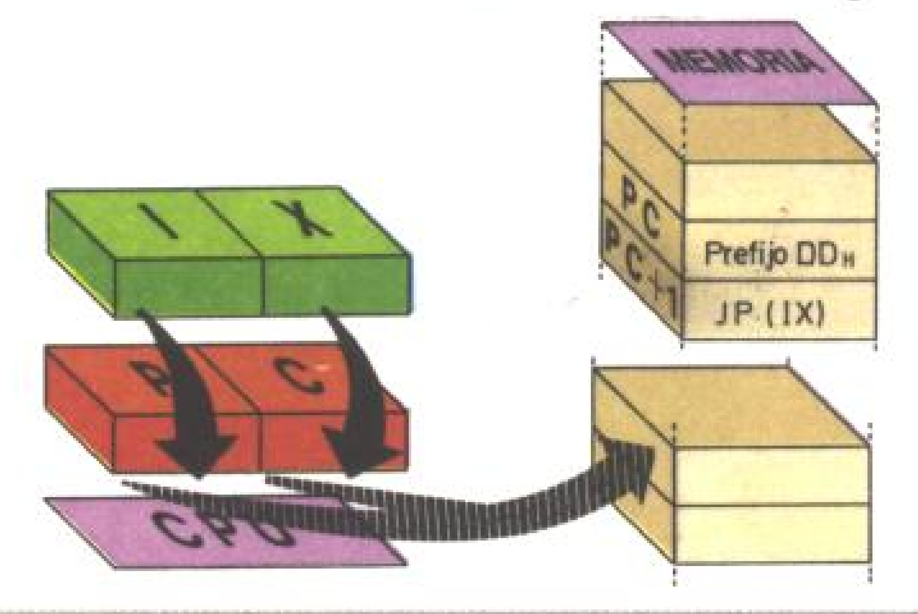
Formato binario:





Ciclos: 2
Estados: 8 (4,4)

Indicadores: ninguno

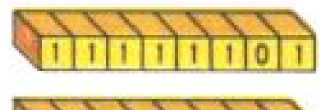


JP (IY)

El contenido del registro índice IY es transferido al registro contador de programa PC, saltando a aquella dirección la ejecución del programa.

Mnemónico: JP **Operandos:** (IY)

Formato binario:



1 1 1 0 1 0 0 1

Ciclos: 2
Estados: 8 (4,4)

Indicadores: ninguno

Ejemplo:

Si el registro índice IY contiene la dirección B316H, después de ejecutar la instrucción:

JP (IY)

el registro PC contendrá B316H. Debido a esto, a continuación no se ejecutará la instrucción siguiente sino la situada en la dirección B316H.

JR e

El operando de desplazamiento «e» es sumado al registro contador de programa PC en el cual queda el resultado, saltando a esta dirección la ejecución del programa.

El operando «e» es un número de 8 bits en complemento a 2, por lo que puede tomar valores de —128 a 127.

Mnemónico: JR **Operandos:** e

Formato binario:

0	0	0	1	1	0	0	0
e-2	e-2	e-2	e-2	e-2	e-2	e-2	e-2

Ciclos: 3
Estados: 12 (4,3,5)

Indicadores: ninguno

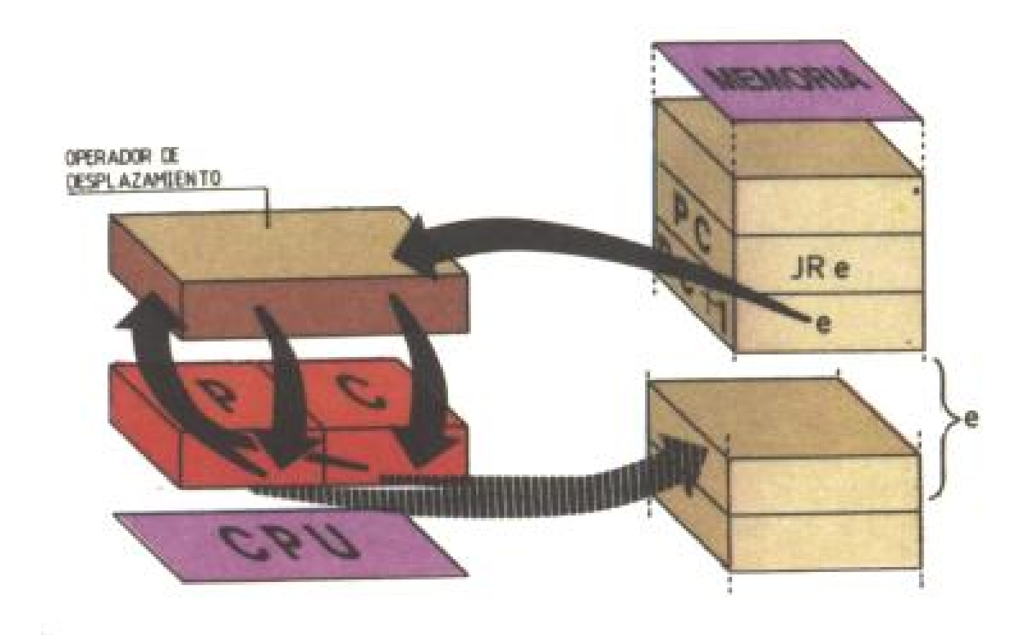
Ejemplo:

Si en las direcciones EA67H y EA68H se encuentra la instrucción:

JR —5

al ejecutar esta instrucción el contador de programa PC contendrá EA69H, que al ser sumado con —5 resultará contener EA64H, ejecutándose a continuación la instrucción situada en esta dirección.

Instr.	Hex.	Dec.
JR e	18,e	24,e
DJNZ e	10,e	16,e



DJNZ e

El registro B es decrementado en la unidad y si el resultado no es 0 termina la instrucción.

Si B—1 resulta ser 0 el operando de desplazamiento «e» es sumado al registro PC en el cual queda el resultado, saltando a esta dirección la ejecución del programa.

El operando «e» es un número de 8 bits en complemento a 2, por lo que puede tomar valores de —128 a 127.

Mnemónico: DJNZ **Operandos:** e

para B < > 0	para B = 0
Ciclos: 3	**Ciclos:** 2
Estados: 13 (5,3,5)	**Estados:** 8 (5,3)

Formato binario: **Indicadores:** ninguno

0	0	0	1	0	0	0	0
e-2	e-2	e-2	e-2	e-2	e-2	e-2	e-2

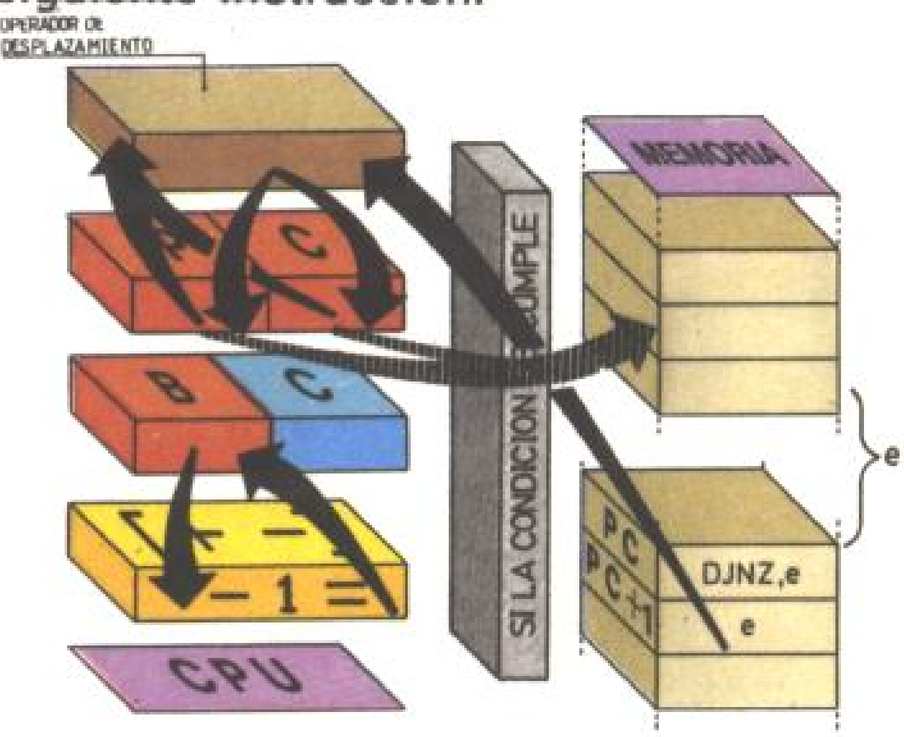
Ejemplo:

Si el registro B contiene 1 y en las direcciones 67A3H y 67A4H se encuentra la instrucción:

DJNZ —8

al ejecutar esta instrucción el registro B contendrá 0 y el contador de programa PC contendrá 67A5H, que al ser sumado con —8 resultará contener 679DH, ejecutándose a continuación la instrucción situada en esta dirección.

Si el registro B contiene cualquier otro valor es decrementado y posteriormente se ejecuta la siguiente instrucción.



JR NZ,e

Si el indicador Z contiene 1 (Z) no se efectúa operación, si contiene 0 (NZ) el operando de desplazamiento «e» es sumado al registro contador del programa PC en el cual queda el resultado, saltando a esta dirección la ejecución del programa.

El operando «e» es un número de 8 bits en complemento a 2, por lo que puede tomar valores de —128 a 127.

Instr.	Hex.	Dec.
JR NZ,e	20,e	32,e
JR Z,e	28,e	40,e
JR NC,e	30,e	48,e
JP C,e	38,e	56,e

Mnemónico: JR

Operandos: NZ,e

Si la condición se cumple

Si la condición no se cumple

Ciclos: 3

Ciclos: 7

Estados: 12 (4,3,5)

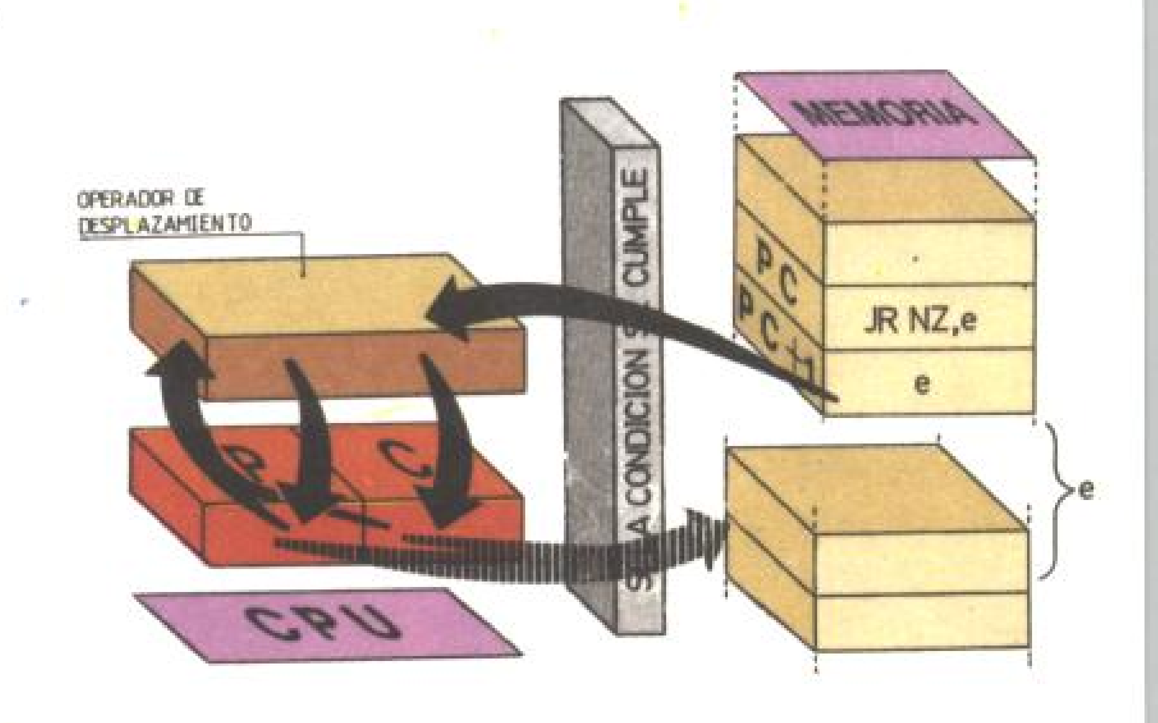
Estados: 7 (4,3)

Formato binario:

Indicadores: ninguno

0 0 1 0 0 0 0 0





JR Z,e

Si el indicador Z contiene 1 el operando de desplazamiento «e» es sumado al registro contador de programa PC en el cual queda el resultado.

Mnemónico: JR
Si se cumple
Ciclos: 3
Estados: 12 (4,3,5)
Formato binario:

0 0 1 0 1 0 0 0

Operandos: Z,e
Si no se cumple
Ciclos: 7
Estados: 7 (4,3)
Indicadores: ninguno



JR NC,e

Si el indicador C contiene 0 el operando de desplazamiento «e» es sumado al registro contador de programa PC en el cual queda el resultado.

Mnemónico: JR
Si se cumple
Ciclos: 3
Estados: 12 (4,3,5)
Formato binario:

0 0 1 1 0 0 0 0

Operandos: NC,e
Si no se cumple
Ciclos: 7
Estados: 7 (4,3)
Indicadores: ninguno



JR C,e

Si el indicador C contiene 1 el operando de desplazamiento «e» es sumado al registro contador del programa PC en el cual queda el resultado.

Mnemónico: JR
Si se cumple
Ciclos: 3
Estados: 12 (4,3,5)
Formato binario:

0 0 1 1 1 0 0 0

Operandos: C,e
Si no se cumple
Ciclos: 2
Estados: 7 (4,3)
Indicadores: ninguno



CALL nn   CALL cc,nn

CALL nn

Primero el contenido del registro contador de pograma PC es almacenado en la pila de máquina: Se decrementa el registro SP, y en la dirección que éste señale se carga el byte más significativo del registro PC, se decrementa de nuevo el registro SP y en la dirección que señale se carga el byte menos significativo de PC.

Posteriormente se carga el registro PC con el número «nn» de 32 bits pasando a ejecutarse la instrucción contenida en esta dirección.

Instr.	Hex.	Dec.
CALL nn	CD,n,n	205,n,n
CALL NZ,nn	C4,n,n	196,n,n
CALL Z,nn	CC,n,n	204,n,n
CALL NC,nn	D4,n,n	212,n,n
CALL C,nn	DC,n,n	220,n,n
CALL PO,nn	E4,n,n	228,n,n
CALL PE,nn	EC,n,n	236,n,n
CALL P,nn	F4,n,n	244,n,n
CALL M,nn	FC,n,n	252,n,n

Mnemónico: CALL   **Operandos:** nn

Formato binario:

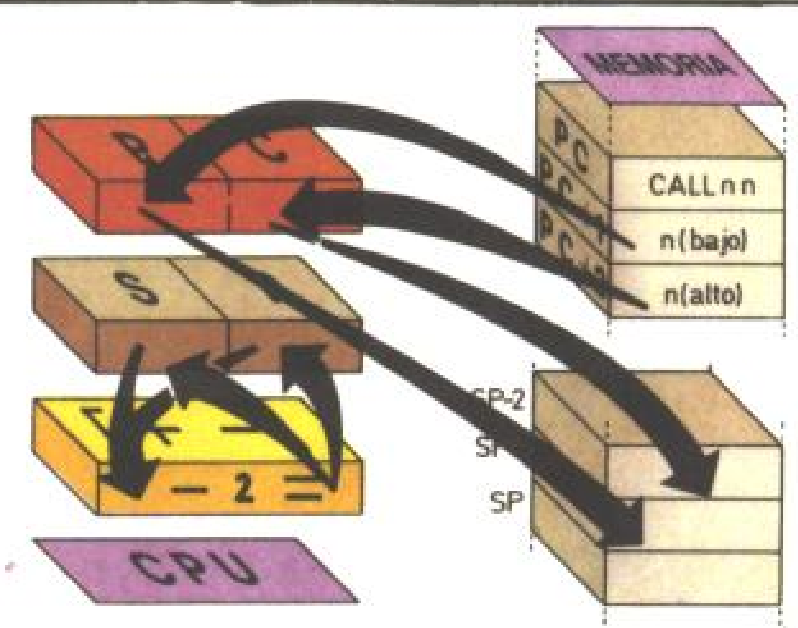
1 1 0 0 1 1 0 1





Ciclos: 5
Estados: 17 (4,3,4,3,3)

Indicadores: ninguno

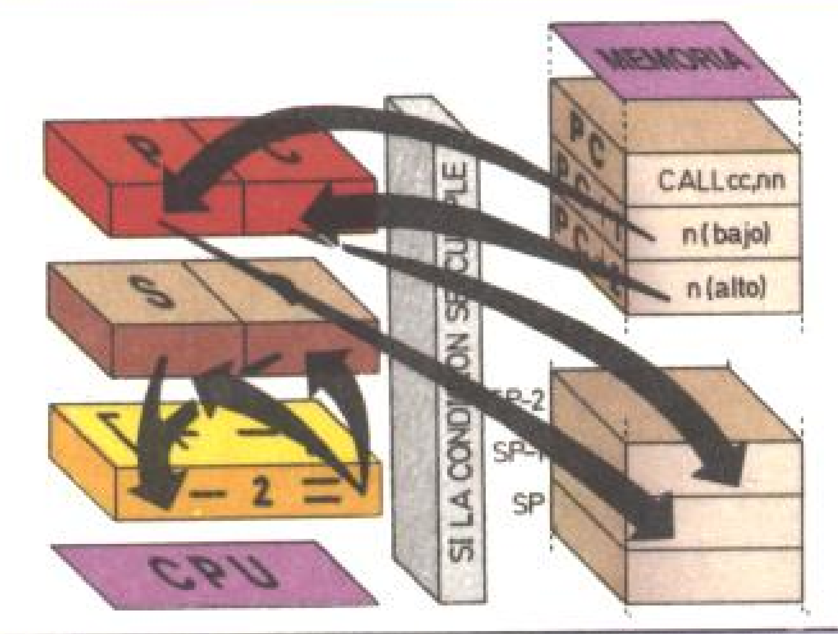


CALL cc,nn

Si la condición «cc» no se cumple no se efectúa ninguna operación y pasa a ejecutarse la instrucción siguiente.

Si se cumple la condición el contenido del contador de programa PC es almacenado en la pila de máquina: Se decrementa el registro SP, y en la dirección que éste señale se carga el byte más significativo de PC, se decrementa de nuevo SP y en la dirección que señale se carga el byte menos significativo de PC.

Posteriormente se carga el registro PC con el número «nn» de 32 bits pasando a ejecutarse la instrucción contenida en esta dirección.



Mnemónico: CALL
Si se cumple
Ciclos: 5
Estados: 17 (4,3,4,3,3)
Formato binario:

Operandos: cc,nn
Si no se cumple
Ciclos: 3
Estados: 10 (4,3,3)
Indicadores: ninguno



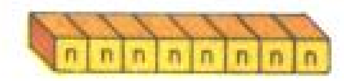




Tabla de Condiciones

cc		Condición	Flag	
000	NZ	no cero	Z	(=0)
001	Z	cero	Z	(=1)
010	NC	no carry	C	(=0)
011	C	carry	C	(=1)
100	PO	paridad impar	P/V	(=0)
101	PE	paridad par	P/V	(=1)
110	P	signo positivo	S	(=0)
111	M	signo negativo	S	(=1)

RET

El último dato almacenado en la pila de máquina es transferido al registro contador del programa PC: Se carga la parte baja del registro PC con el contenido de la dirección especificada por el registro SP, se incrementa el par SP, se carga la parte alta del registro PC de la misma manera y se vuelve a incrementar el registro SP.

Posteriormente pasa a ejecutarse la instrucción contenida en la dirección cargada en el contador de programa PC.

Instr.	Hex.	Dec.
RET	C9	201
RET NZ	C0	192
RET Z	C8	200
RET NC	D0	208
RET C	D8	216
RET PO	E0	224
RET PE	E8	232
RET P	F0	240
RET M	F8	248

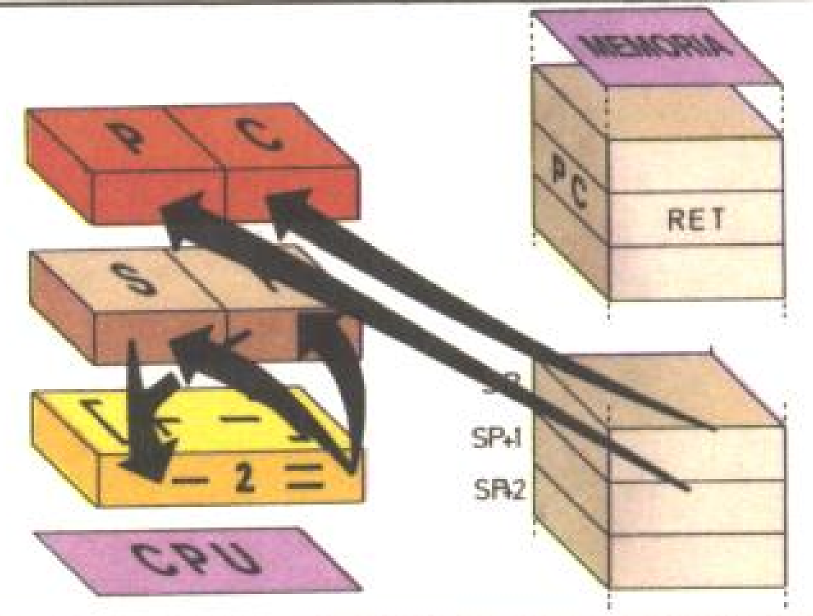
Mnemónico: RET **Operandos:** no tiene

Formato binario:

1 1 0 0 1 0 0 1

Ciclos: 3
Estados: 10 (4,3,3)

Indicadores: ninguno



RET cc

Si la condición «cc» no se cumple no se efectúa ninguna operación y pasa a ejecutarse la instrucción siguiente.

Si se cumple la condición el último dato almacenado en la pila de máquina es transferido al registro contador de programa PC: Se carga la parte baja del registro P C con el contenido de la dirección especificada por SP, se incrementa el par SP, se carga la parte alta de PC de la misma manera y se vuelve a incrementar SP.

Posteriormente pasa a ejecutarse la instrucción contenida en la dirección cargada en el contador de programa PC.

Mnemónico: RET
Si se cumple
Ciclos: 3
Estados: 11 (5,3,3)
Formato binario:



Operandos: cc
Si no se cumple
Ciclos: 1
Estados: 5
Indicadores: ninguno

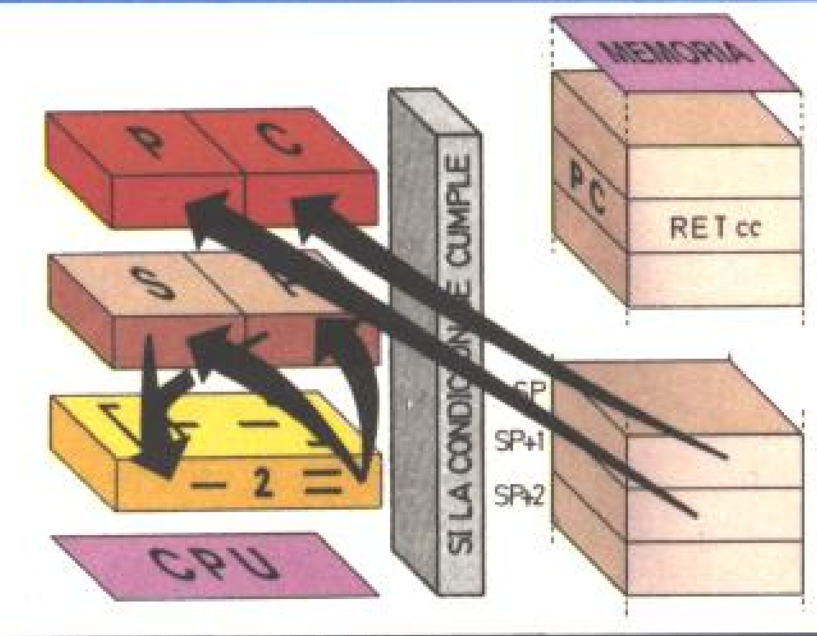


Tabla de Condiciones

cc	Condición		Flag	
000	NZ	no cero	Z	(=0)
001	Z	cero	Z	(=1)
010	NC	no carry	C	(=0)
011	C	carry	C	(=1)
100	PO	paridad impar	P/V	(=0)
101	PE	paridad par	P/V	(=1)
110	P	signo positivo	S	(=0)
111	M	signo negativo	S	(=1)

RST P RETI RETN

RST p

Primero el contenido del registro contador de programa PC es almacenado en la pila de máquina: Se decrementa el registro SP, y en la dirección que éste señale se carga el byte más significativo del registro PC, se decrementa de nuevo el registro SP y en la dirección que señale se carga el byte menos significativo de PC.

Posteriormente se carga la parte alta del registro PC con 0 y la parte baja de éste con el operando «p» de 8 bits.

Instr.	Hex.	Dec.
RST 0H	C7	199
RST 8H	CF	207
RST 10H	D7	215
RST 18H	DF	223
RST 20H	E7	231
RST 28H	EF	239
RST 30H	F7	247
RST 38H	FF	255
RETI	ED,4D	237,77
RETN	ED,45	237,69

Mnemónico: RST **Operandos:** p

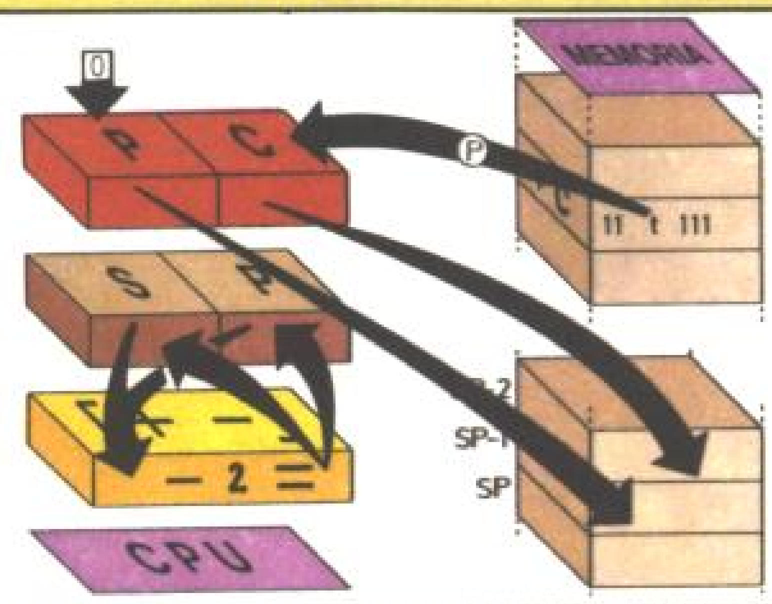
Formato binario:



Ciclos: 3

Estados: 11 (5,3,3)

Indicadores: ninguno



Direcciones de RESTART:			
t	p	t	p
000	0000H	100	0020H
001	0008H	101	0028H
010	0010H	110	0030H
011	0018H	111	0038H

RETI

Retorno de una interrupción enmascarable: El último dato almacenado en la pila de máquina es transferido al registro contador de programa PC al igual que en la instrucción RET. Los dispositivos periféricos son informados de que ha finalizado la rutina de servicio de interrupción.

Mnemónico: RETI **Operandos:** no tiene

Formato binario:

1 1 1 0 1 1 0 1

0 1 0 0 1 1 0 1

Ciclos: 4
Estados: 14 (4,4,3,3)

Indicadores: ninguno

RETN

Retorno de una interrupción no enmascarable: El último dato almacenado en la pila de máquina es transferido al registro contador de programa PC al igual que en la instrucción RET; además la báscula de interrupción IFF2 es copiada en IFF1.

Mnemónico: RETN **Operandos:** no tiene

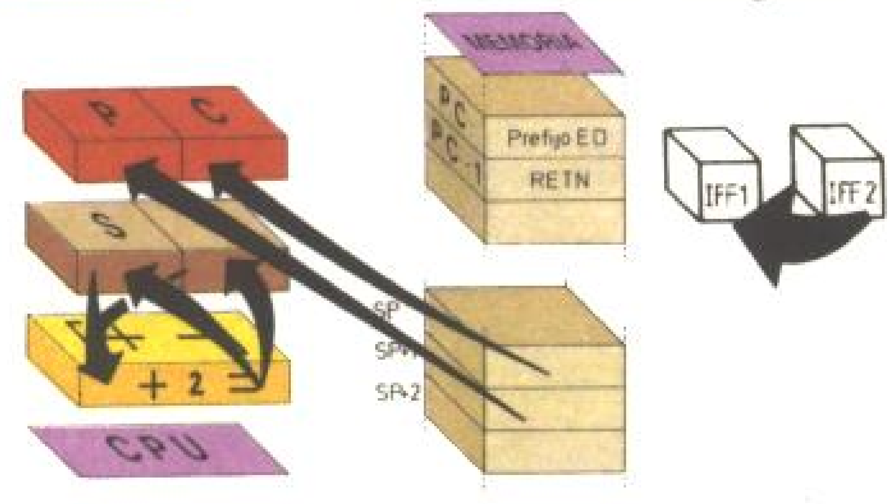
Formato binario:

1 1 1 0 1 1 0 1

0 1 0 0 0 1 0 1

Ciclos: 4
Estados: 14 (4,4,3,3)

Indicadores: ninguno



IN A,(N)

El número de dispositivo «n» de 8 bits es colocado en la parte baja del bus de direcciones y el contenido del Acumulador en la parte alta del mismo. Es leído un byte por el puerto seleccionado y cargado en el registro A.

Mnemónico: IN   **Operandos:** A,(n)

Formato binario:

1	1	0	1	1	0	1	1
n	n	n	n	n	n	n	n

Ciclos: 3
Estados: 11 (4,3,4)

Indicadores: ninguno

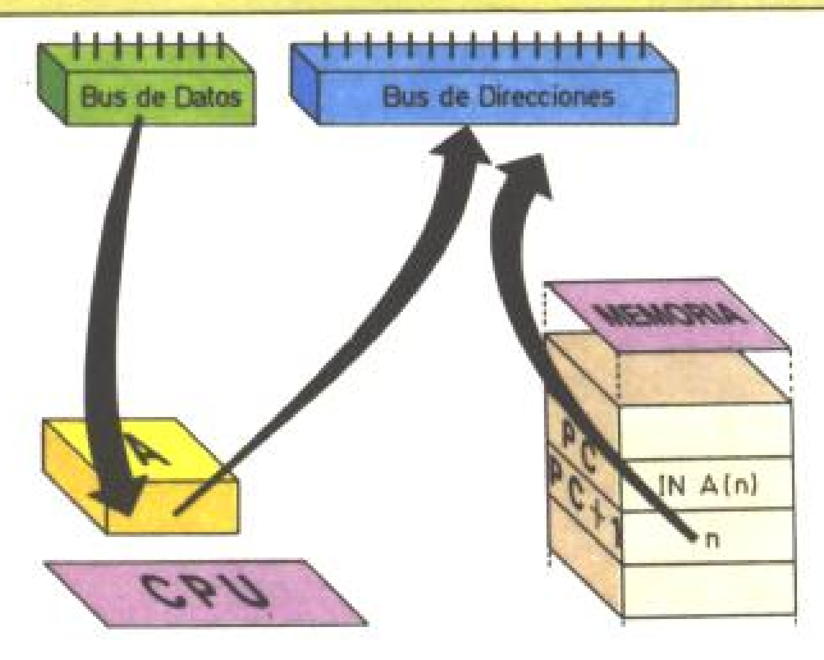
Ejemplo

Si el registro A contiene DFH, después de la instrucción:

IN A,(FEH)

El valor de 8 bits depositado por el periférico conectado al puerto FEH (teclado) correspondiente a la semifila DFH (YUIOP) será cargado en el acumulador.

Instr.	Hex.	Dec.
IN A,(n)	DB,n	219,n
IN A,(C)	ED,78	237,120
IN B,(C)	ED,40	237,64
IN C,(C)	ED,48	237,72
IN D,(C)	ED,50	237,80
IN E,(C)	ED,58	237,88
IN H,(C)	ED,60	237,96
IN L,(C)	ED,68	237,104



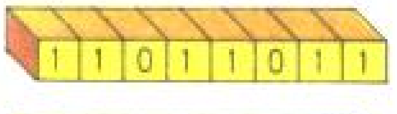
IN r,(C)

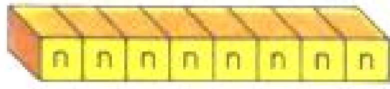
El número de dispositivo contenido en el registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B en la parte alta del mismo. Es leído un byte por el puerto seleccionado y cargado en el registro «r» determinado por la instrucción.

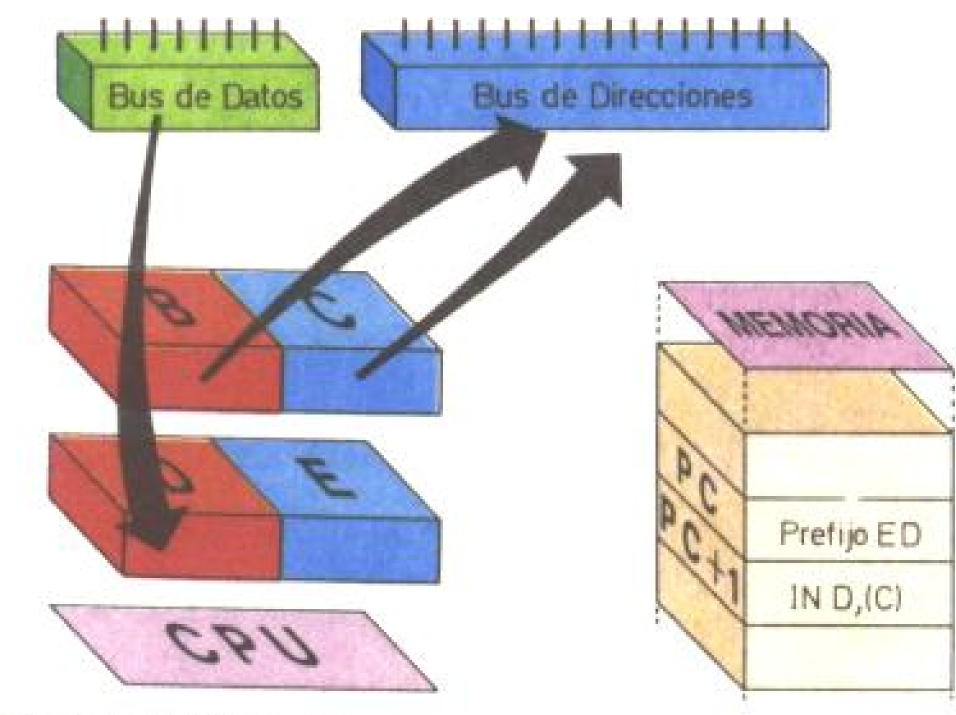
Mnemónico: IN **Operandos:** r,(C)

Formato binario:

Ciclos: 3
Estados: 12 (4,4,4)







Indicadores:

S a 1 si el dato de entrada es negativo.
Z a 1 si el dato de entrada es 0.
H a 0.
P/V a 1 si el dato de entrada tiene paridad par.
N a 0.
C no afectado.

Observaciones:

El código ED,70H (237,112d) tiene el mismo formato que las instrucciones IN r,(C) pero no corresponde a ningún registro, por lo que no tiene mnemónico asociado, no obstante, esta instrucción, funciona colocando los indicadores aunque el dato no es cargado en ningún registro.

INI

El contenido del registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B en la parte alta del mismo. Es leído un byte del puerto seleccionado y cargado en la posición de memoria especificada por el contenido del par HL. Posteriormente el par HL es incrementado.

El registro B es decrementado, lo que permite utilizarlo como contador en un bucle de INIs sucesivos.

Mnemónico: INI **Operandos:** no tiene

Formato binario:

1 1 1 0 1 1 0 1

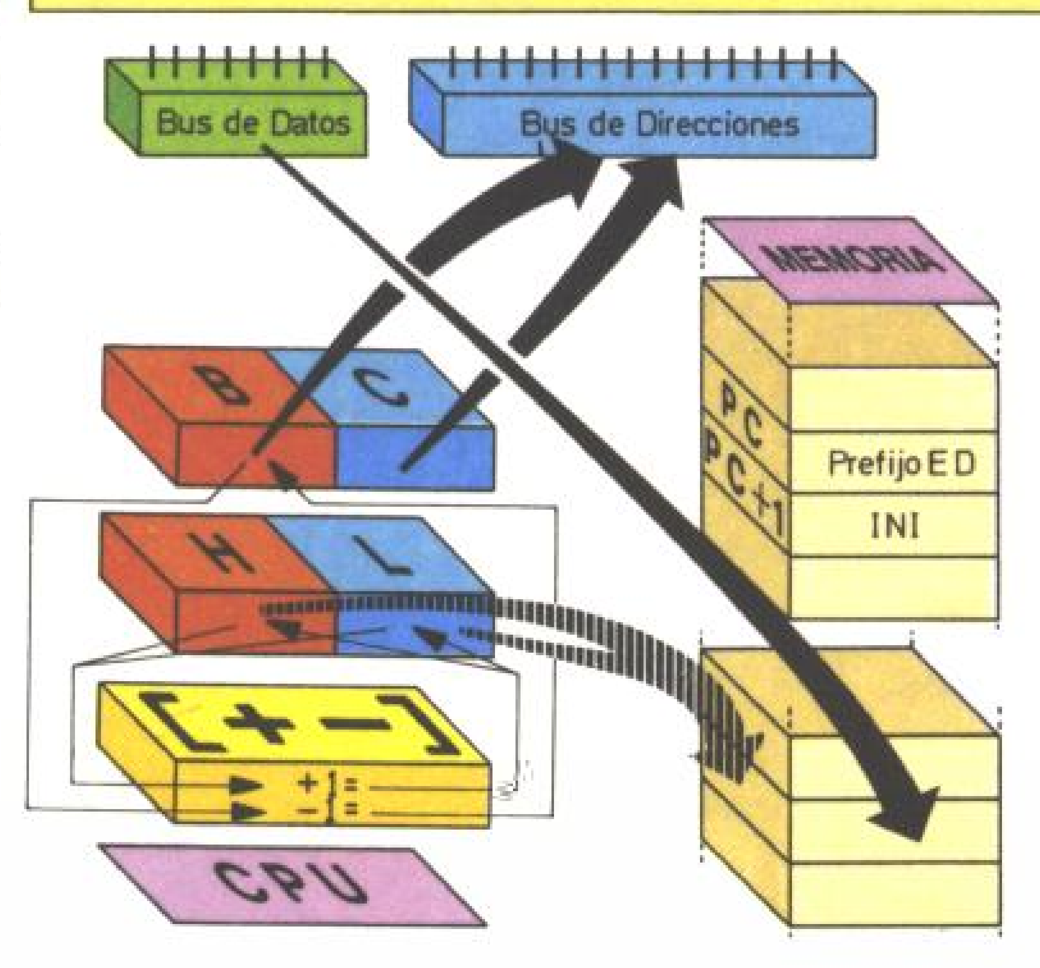
1 0 1 0 0 0 1 0

Ciclos: 4
Estados: 16 (4,5,3,4)

Indicadores:

S desconocido
Z A 1 si B-1 resulta 0
H desconocido
P/V desconocido
N a 1
C no afectado

Instr.	Hex.	Dec.
INI	ED,A2	237,162
INIR	ED,B2	237,178



INIR

Se repite la secuencia INI hasta que el registro B resulte 0, en cuyo caso termina la instrucción.

Por lo tanto, se transfiere al contenido de un bloque de memoria que comienza en la dirección señalada por el par HL, la cantidad de información determinada por el registro B procedente del periférico conectado al puerto especificado por el registro C.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: INIR	**Operandos:** no tiene
para BC < > Ø	para BC = Ø
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,5,3,4,5)	**Estados:** 16 (4,5,3,4)

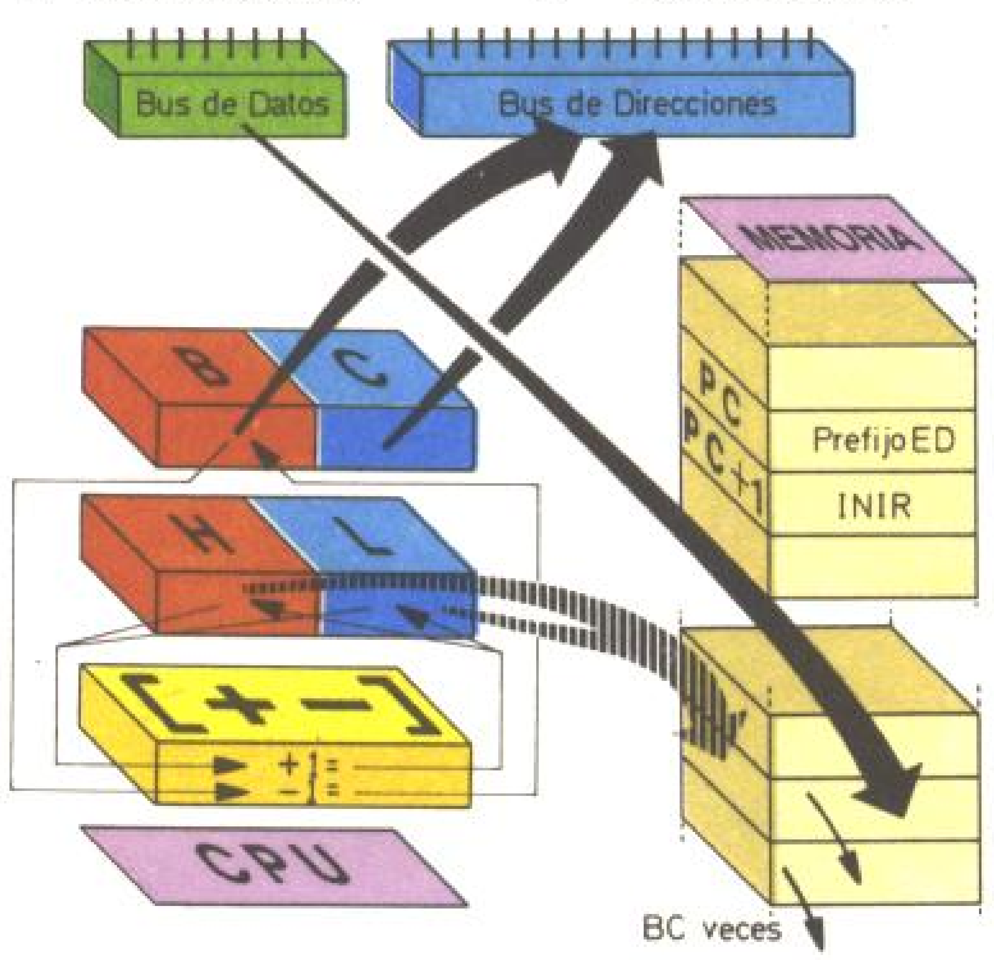
Formato binario:

1 1 1 0 1 1 0 1

1 0 1 1 0 0 1 0

Indicadores:

S desconocido — P/V desconocido
Z a 1 — N a 1
H desconocido — C no afectado.



IND

El contenido del registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B en la parte alta del mismo. Es leído un byte del puerto seleccionado y cargado en la posición de memoria especificada por el contenido del par HL. Posteriormente el par HL es decrementado.

El registro B es decrementado, lo que permite utilizarlo como contador en un bucle de INDs sucesivos.

Mnemónico: IND **Operandos:** no tiene

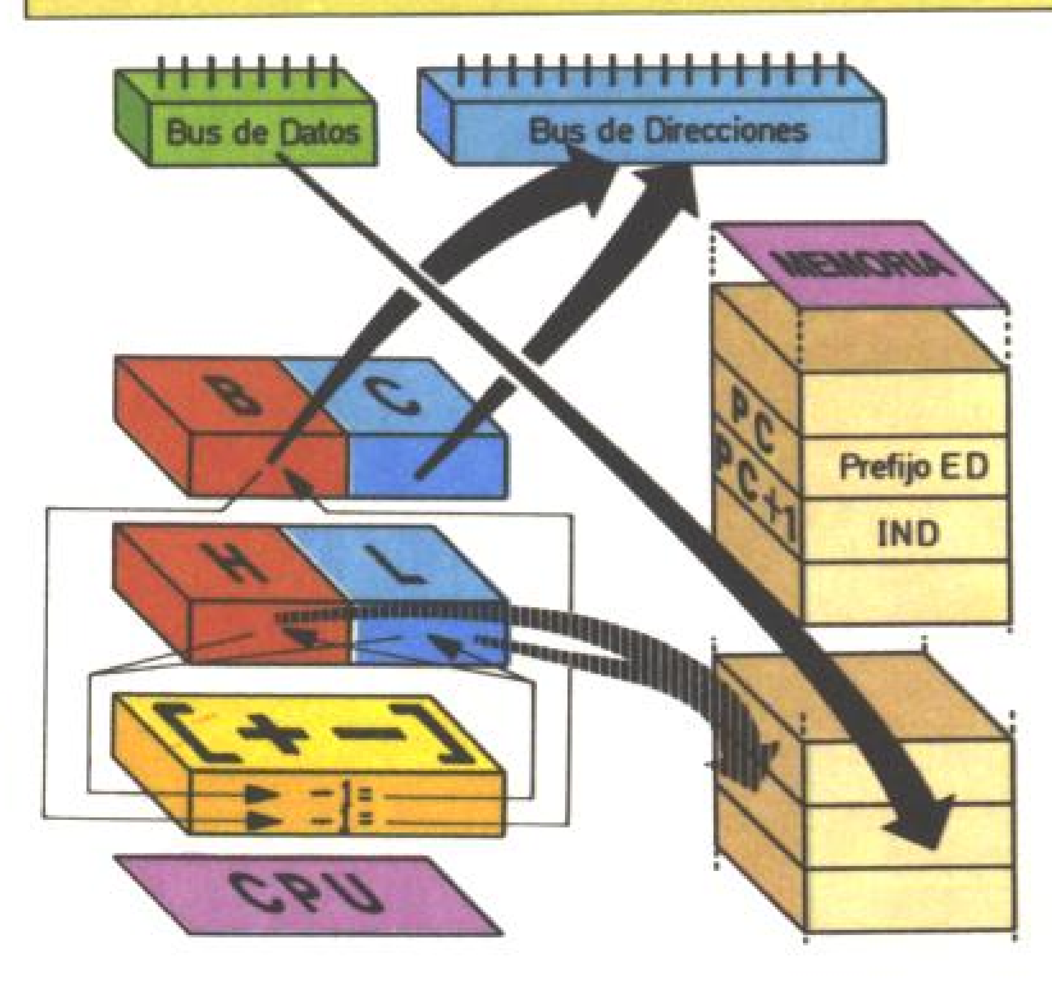
Formato binario:

11101101

10101010

Ciclos: 4
Estados: 16 (4,5,3,4)
Indicadores:
S desconocido
Z A 1 si B-1 resulta 0
H desconocido
P/V desconocido
N a 1
C no afectado

Instr.	Hex.	Dec.
IND	ED,AA	237,170
INDR	ED,BA	237,186



INDR

Se repite la secuencia IND hasta que el registro B resulte 0, en cuyo caso termina la instrucción.

Por lo tanto, se transfiere al contenido de un bloque de memoria que termina en la dirección señalada por el par HL, la cantidad de información determinada por el registro B procedente del periférico conectado al puerto especificado por el registro C.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: LDDR **Operandos:** no tiene

para BC < > Ø	para BC = Ø
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,5,3,4,5)	**Estados:** 16 (4,5,3,4)

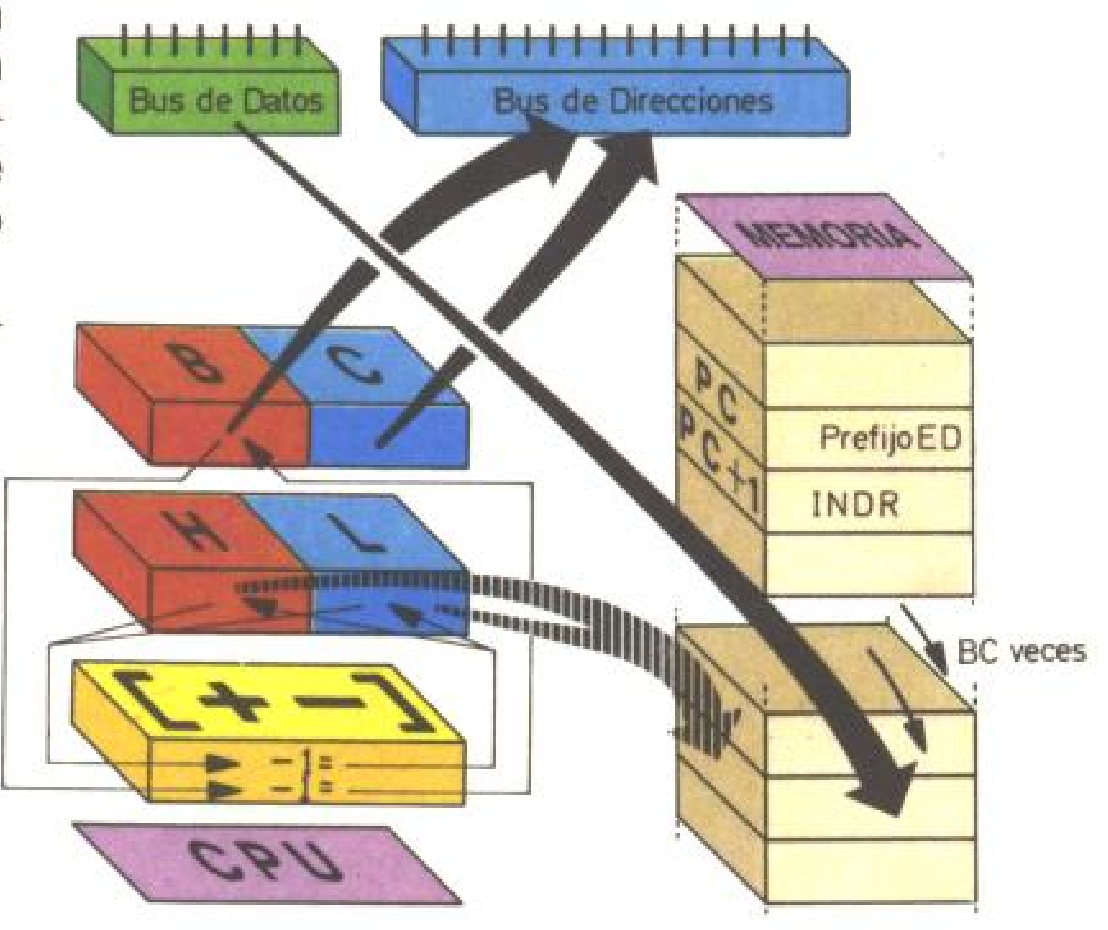
Formato binario:

1 1 1 0 1 1 0 1

1 0 1 1 1 0 1 0

Indicadores:

S	desconocido	P/V	desconocido
Z	a 1	N	a 1
H	desconocido	C	no afectado.



OUT (n),A   OUT (C),r

OUT (N),A

El número de dispositivo «n» de 8 bits es colocado en la parte baja del bus de direcciones y el contenido del Acumulador en la parte alta de éste y, al mismo tiempo, en el bus de datos. De esta forma el contenido del acumulador es transferido al periférico determinado por el operando «n»

Mnemónico: OUT   **Operandos:** (n),A

Formato binario:

1	1	0	1	0	0	1	1
n	n	n	n	n	n	n	n

Ciclos: 3
Estados: 11 (4,3,4)

Indicadores: ninguno

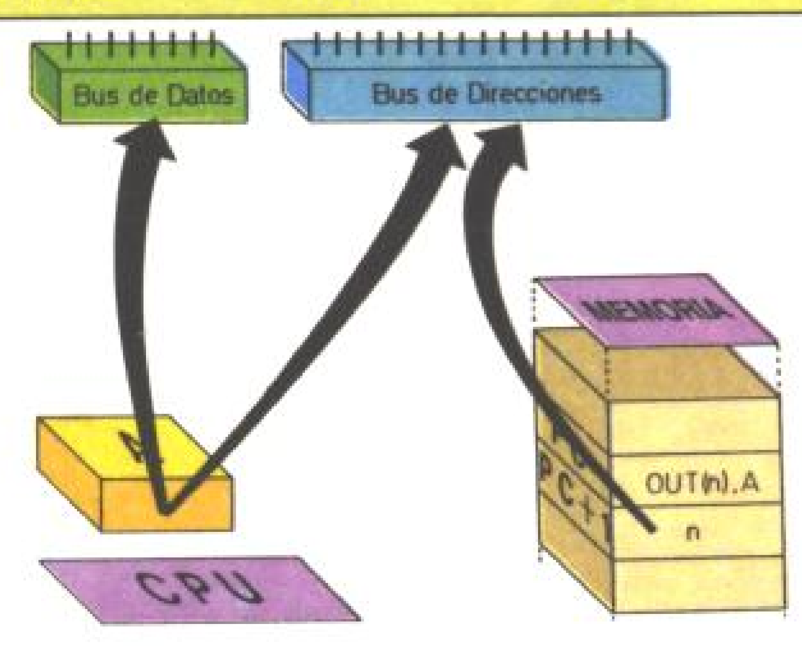
Ejemplo:

Si el registro A contiene 02H, después de la instrucción.

OUT (FEH),A

El valor 02H es depositado en el periférico FEH (BORDER) por lo que el borde de la pantalla aparecerá de color rojo.

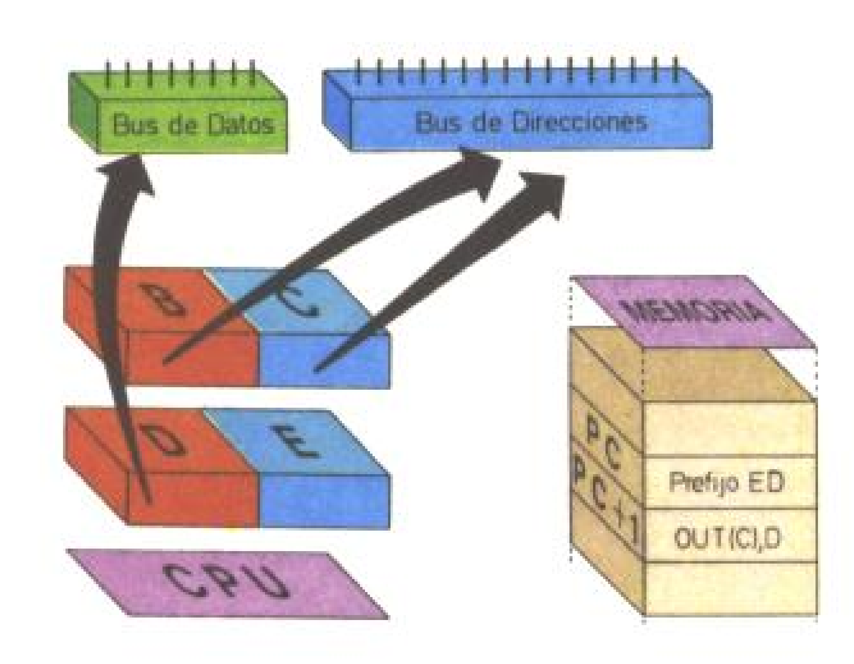
Instr.	Hex.	Dec.
OUT (n),A	D3,n	211,n
OUT (C),A	ED,79	237,121
OUT (C),B	ED,41	237,65
OUT (C),C	ED,49	237,73
OUT (C),D	ED,51	237,81
OUT (C),E	ED,59	237,89
OUT (C),H	ED,61	237,97
OUT (C),L	ED,69	237,105



OUT (C),r

El número de dispositivo contenido en el registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B en la parte alta del mismo.

El contenido del registro «r» determinado por la instrucción es depositado en el bus de datos para ser recibido por el periférico conectado al puerto indicado.



Mnemónico: OUT **Operandos:** (C),r

Formato binario:

1	1	1	0	1	1	0	1
0	1	r	r	r	0	0	1

Ciclos: 3

Estados: 12 (4,4,4)

Indicadores: ninguno.

Ejemplo:

Si el registro H contiene 05H y el registro C contiene FEH, después de la instrucción.

OUT (C),H

El valor 05H es depositado en el periférico FEH (BORDER) por lo que el borde de la pantalla aparecerá de color azul claro.

Instr.	Hex.	Dec.
OUTI	ED,A3	237,163
OTIR	ED,B3	237,179

OUTI

El contenido del registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B -1 en la parte alta del mismo. En el bus de datos es escrito el contenido de la posición de memoria especificada por el par de registros HL para ser enviado al periférico correspondiente. Posteriormente el par HL es incrementado.

El registro B es decrementado, lo que permite utilizarlo como contador en un bucle de OUTIs sucesivos.

Mnemónico: OUTI **Operandos:** no tiene

Formato binario:

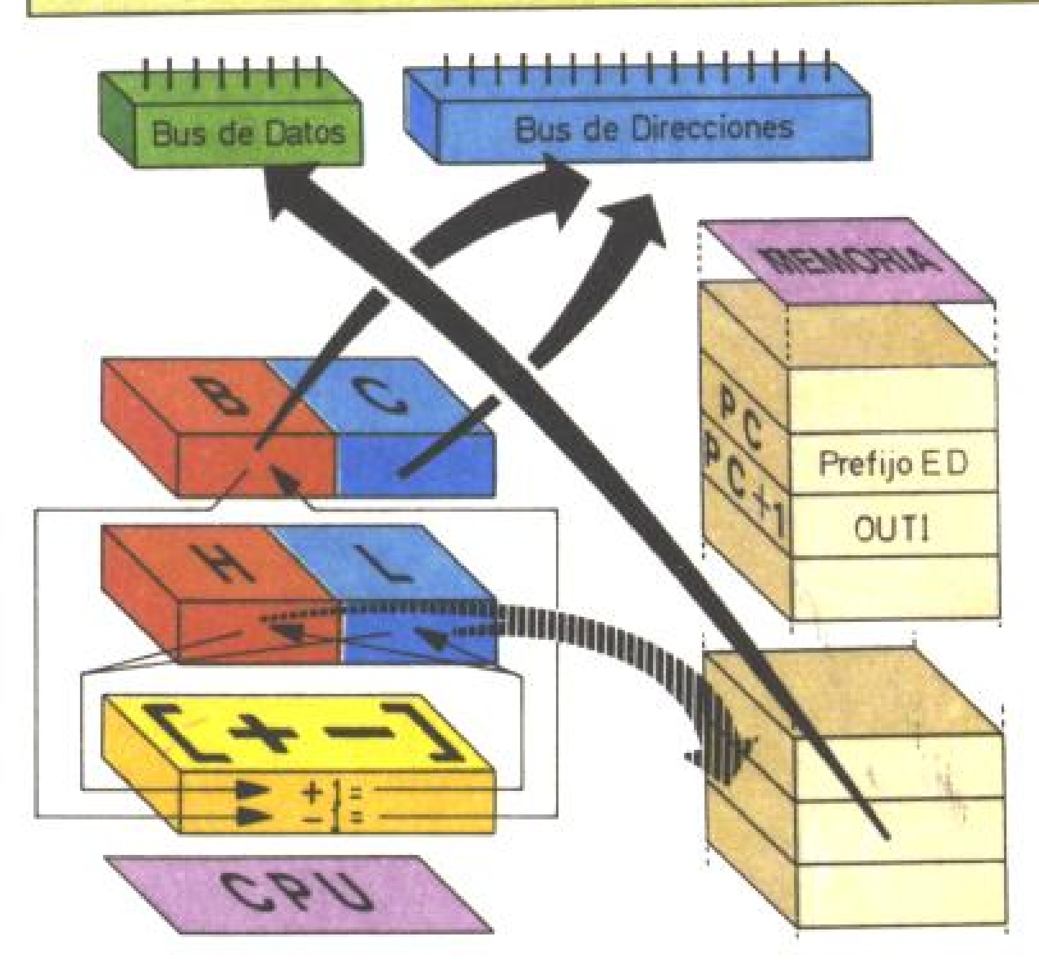
1 1 1 0 1 1 0 1

1 0 1 0 0 0 1 1

Ciclos: 4
Estados: 16 (4,5,3,4)

Indicadores:

S desconocido
Z A 1 si B-1 resulta 0
H desconocido
P/V desconocido
N a 1
C no afectado



OTIR

Se repite la secuencia OUTI hasta que el registro B resulte 0, en cuyo caso termina la instrucción.

Por lo tanto, se transfiere el contenido de un bloque de memoria que comienza en la dirección señalada por el par HL, la cantidad de información determinada por el registro B por el puerto especificado por el registro C al periférico correspondiente.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: OTIR **Operandos:** no tiene

para BC < > 0	para BC = 0
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,5,3,4,5)	**Estados:** 16 (4,5,3,4)

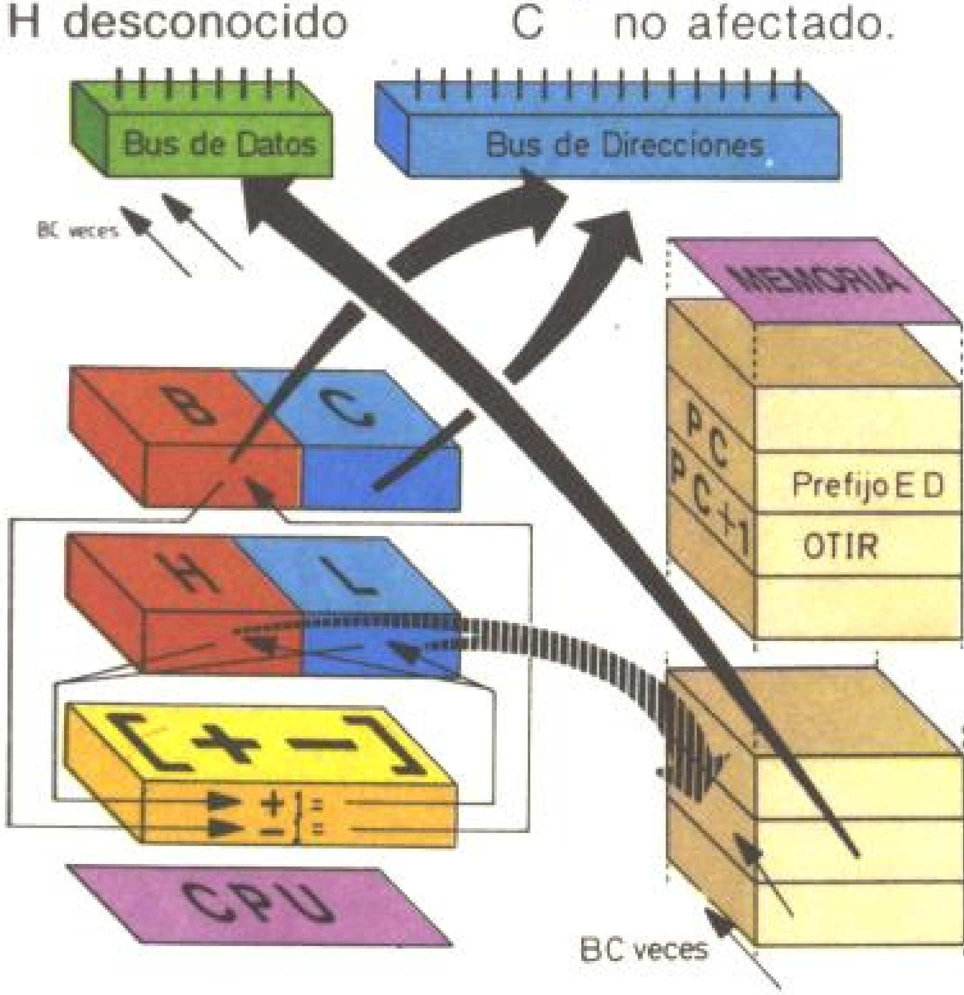
Formato binario:

1 1 1 0 1 1 0 1

1 0 1 1 0 0 1 1

Indicadores:

S	desconocido	P/V	desconocido
Z	a 1	N	a 1
H	desconocido	C	no afectado.



Instr.	Hex.	Dec.
OUTD	ED,AB	237,171
OTDR	ED,BB	237,187

OUTD

El contenido del registro C es colocado en la parte baja del bus de direcciones y el contenido del registro B -1 en la parte alta del mismo. En el byte de datos es escrito el contenido de la posición de memoria especificada por el par de registros HL para ser enviado al periférico correspondiente. Posteriormente el par HL es decrementado.

El registro B es decrementado, lo que permite utilizarlo como contador en un bucle de OUTDs sucesivos.

Mnemónico: OUTD **Operandos:** no tiene

Formato binario:

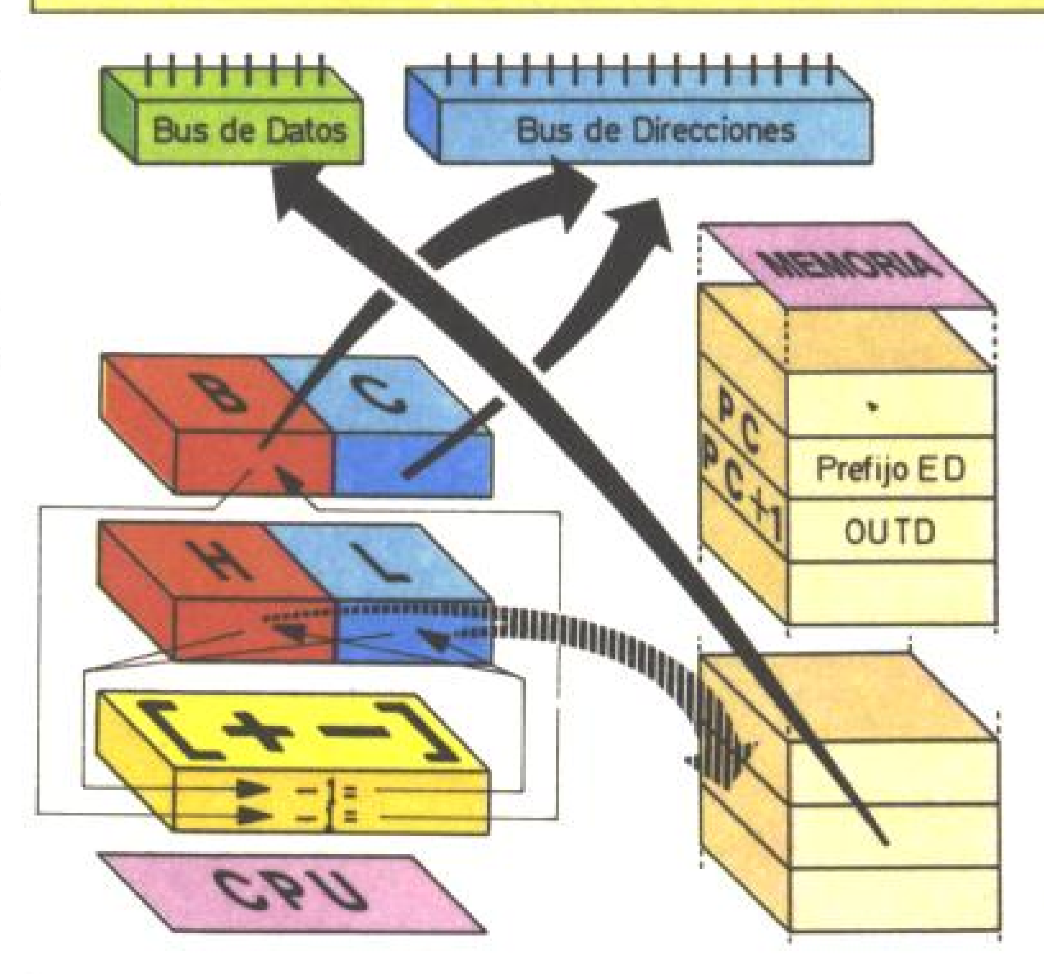
1 1 1 0 1 1 0 1

1 0 1 0 1 0 1 1

Ciclos: 4
Estados: 16 (4,5,3,4)

Indicadores:

S desconocido
Z A 1 si B-1 resulta 0
H desconocido
P/V desconocido
N a 1
C no afectado



OTDR

Se repite la secuencia OUTD hasta que el registro B resulte 0, en cuyo caso termina la instrucción.

Por lo tanto, se transfiere el contenido de un bloque de memoria que termina en la dirección señalada por el par HL, la cantidad de información determinada por el registro B por el puerto especificado por el registro C al periférico correspondiente.

Las peticiones de interrupción son comprobadas al final de cada transferencia.

Mnemónico: OTDR **Operandos:** no tiene

para BC < > Ø	para BC = Ø
Ciclos: 5	**Ciclos:** 4
Estados: 21 (4,5,3,4,5)	**Estados:** 16 (4,5,3,4)

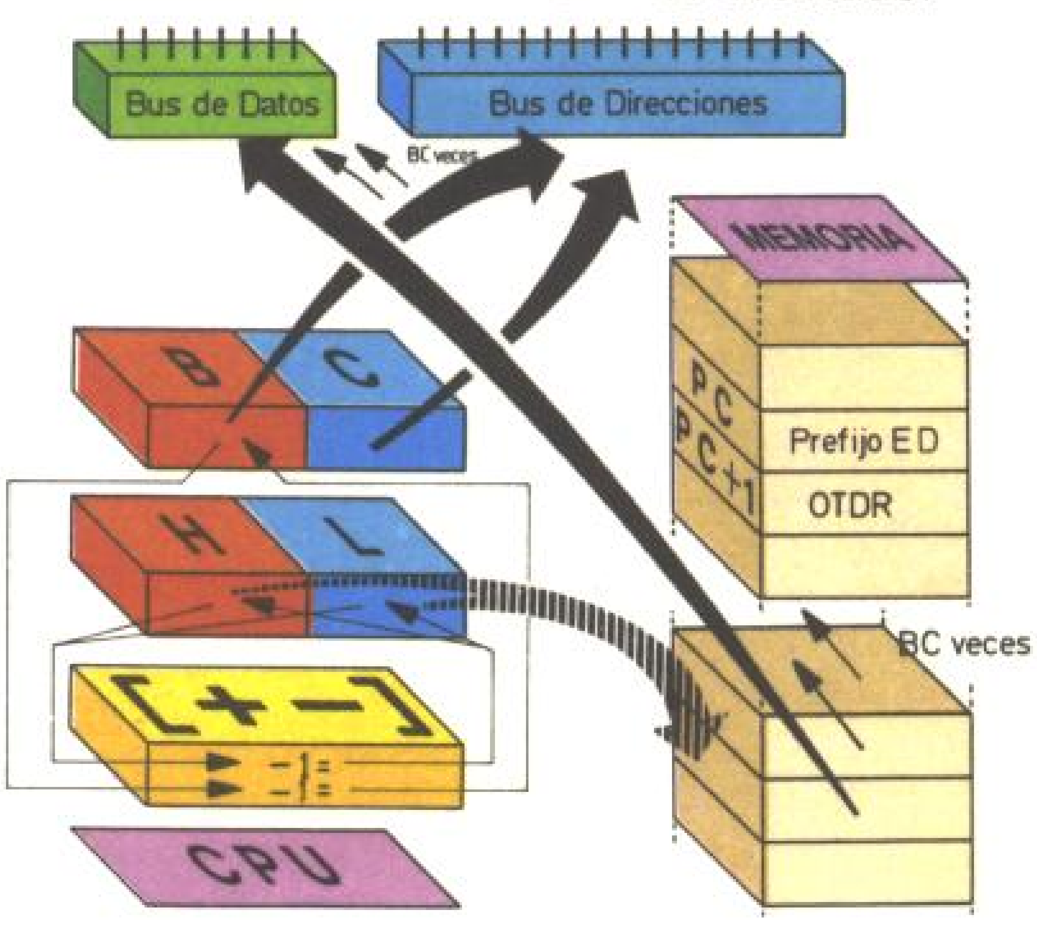
Formato binario:

1 1 1 0 1 1 0 1

1 0 1 1 1 0 1 1

Indicadores:

S desconocido
Z a 1
H desconocido
P/V desconocido
N a 1
C no afectado.



Rutinas de la ROM

La ROM (Memoria de solo lectura) del SPECTRUM consta de 16 K (16384 bytes) entre los que se pueden distinguir:

• Una primera parte la constituyen las rutinas de iniciacilización, y las relativas a los periféricos: Teclado (028EH),sonido (03B5H), cassette (04C2H), y pantalla e impresora (09F4H).

• El bucle principal(12A2H) consiste básicamente en una rutina cíclica que entra en el editor(0F2CH) y en la rutina de ejecución alternativamente.

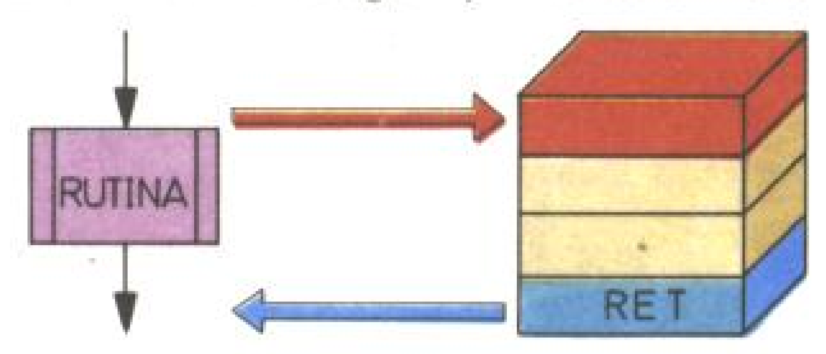
• La rutina de ejecución (1B8AH) recorre el programa ejecutando cada una de las instrucciones.

• La rutina de evaluación de expresiones (24FBH) que tiene un doble funcionamiento según se esté en modo edición o modo ejecución.

• Las rutinas aritméticas independientes (2D4FH) y las efectuadas por el calculador.

• La tabla de caracteres (3D00H) donde se encuentra la definición de todos ellos.

Los nombres de rutinas generales van escritos en MAYUSCULAS, los que aparecen en minúscula corresponden a las rutinas del CALCULADOR (RST 28H). Tanto unos como otros han sido tomados del libro *SPECTRUM ROM DISASSEMBLY»* de Ian Logan y Frank O'Hara.



La tabla de sintaxis que aparece en la microficha T-8 muestra las direcciones de las rutinas de los comandos BASIC. Normalmente estas rutinas **no pueden usarse desde código máquina** pues exigen parámetros escritos en BASIC. Para utilizar estas rutinas desde código máquina (aquéllas que tiene sentido hacerlo) debe hacerse una llamada a la segunda parte de éstas. Las direcciones y la forma correcta de utilizarse se ofrece en las diferentes fichas de esta serie **M**.

Registros

Al entrar en una rutina USR hay que tener en cuenta estos tres registros.

IY contiene la dirección 23610 para permitir manejar las variables del sistema de forma indexada. A menos que se desee engañar a la ROM con una falsa tabla de variables debe restablecerse su valor cada vez que se llame a una rutina que las utilice. (La mayoría).

Al retornar al BASIC no es necesario recuperarla, pues lo hace el sistema.

HL' contiene la dirección de retorno a la rutina SCANNING una vez vuelto al BASIC. Puede usarse sin ningún problema siempre que se restablezca su valor antes de volver al BASIC (2758H = 10072d).

SP contiene la dirección de la pila de máquina y debe contener al volver al BASIC el mismo valor que tenía al salir de él salvo que se pretenda intervenir especialmente (ej.: rutina ON ERROR GOTO).

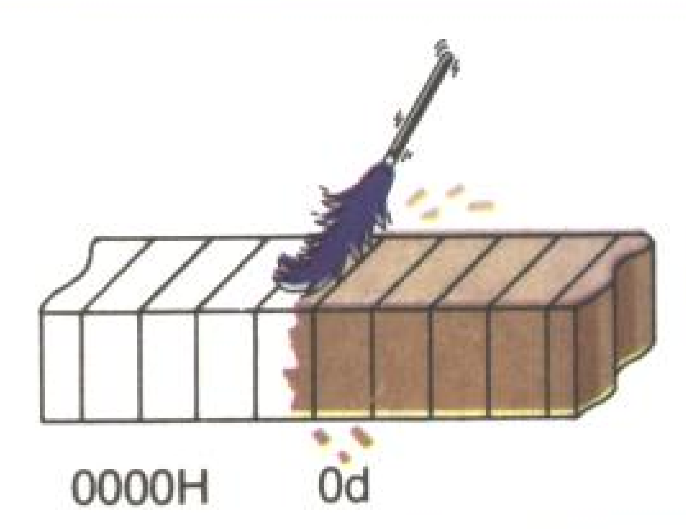
Interrupciones

Durante las interrupciones se pueden usar rutinas de la ROM pero con ciertas precauciones:

- No puede usarse el stack del calculador y por tanto ninguna de las rutinas del CALCULADOR (RST 28H) si el programa principal lo usa (por supuesto el BASIC lo hace). La razón de este impedimento es que en el momento de ser llamada la interrupción se puede estar escribiendo o leyendo un dato.

- Es peligroso mover partes del programa de su lugar pues éstas podrían estar ejecutándose; por lo tanto, no deben llamarse rutinas como MAKE-ROM y RECLAIM ni otras que las usen.

- No debe llamarse a ninguna rutina que cambie variables del sistema si el programa principal es BASIC o usa alguna de éstas (Ejemplo: en lugar de usar RST 10H para escribir en pantalla, debe usarse PO-CHAR (0B65H), que no modifica las variables del sistema.

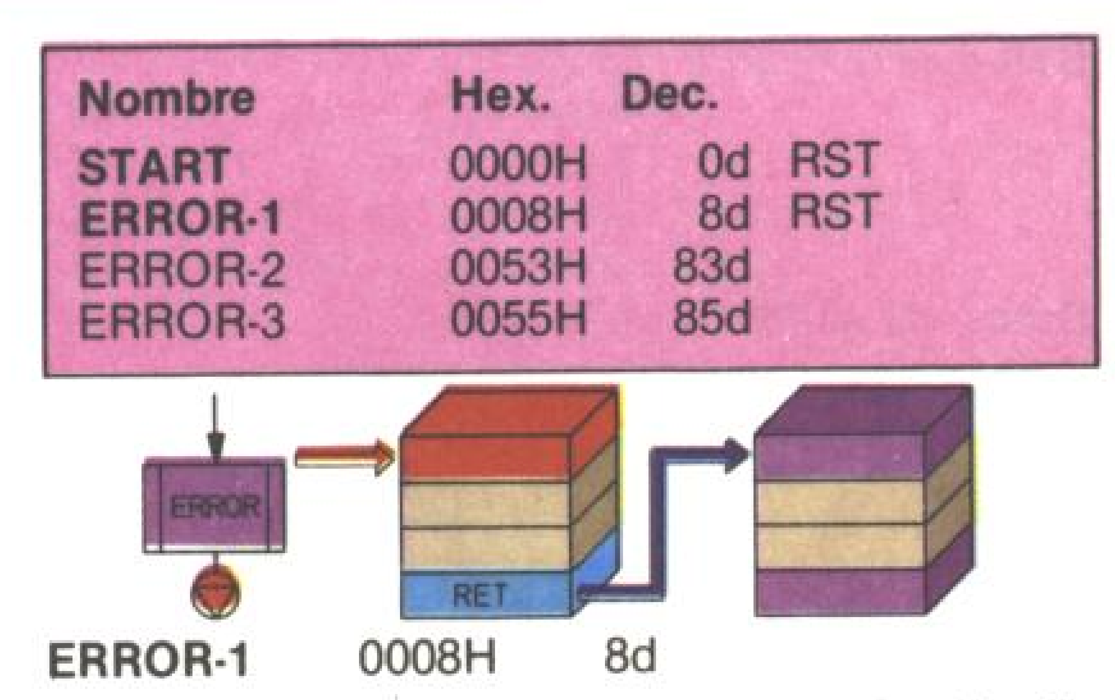


START 0000H 0d

Rutina de inicialización. Es la primera que ejecuta el microprocesador al ser conectado o ejecutar un Reset. Llama a la rutina situada en la dirección 11CBH para comprobar la memoria e inicializar ésta, la pantalla, las variables del sistema y el área de gráficos definidos por el usuario (UDG).

Datos de entrada: Ninguno.
Datos de salida : Memoria inicializada.

Registros modificados: Todos.
Variables modificadas: Todas.
Rutinas que utiliza : START/NEW (11CBH).



Nombre	Hex.	Dec.	
START	0000H	0d	RST
ERROR-1	0008H	8d	RST
ERROR-2	0053H	83d	
ERROR-3	0055H	85d	

ERROR-1 0008H 8d

Rutina de error. Se ejecuta cuando el intérprete Basic ha detectado un error en el programa. Sitúa en X-PTR la dirección del error y en ERR-NR el código de éste menos 1, posteriormente restablece el Stack en (ERR-SP), elimina el stack del calculador y asigna a MEM la dirección de MEMBOT (5C92H). Por último «retorna» a la dirección señalada indirectamente por (ERR-SP), normalmente **MAIN-4** (4867H,1303d) que termina saltando al editor Basic.

Datos de entrada: Código de error menos 1 en el Byte siguiente a RST 8H. Dirección de la rutina de error en la dirección señalada indir. por (ERR-SP).
Datos de salida : SP = (ERRSP), HL = (STKEND)

Registros modificados: HL, SP.
Variables modificadas : X-PTR, ERR-NR, STKEND, MEM

Rutinas que utiliza: **ERROR-2** (0053H), **SET-STK** (16C5H), La rutina de error señalada indir. por (ERR-SP).
Rutina usada por : Gran parte de las rutinas ejecutivas y la mayoría de las numéricas.

Observaciones: Esta rutina debido a que restablece el Stack no retorna a la dirección de donde partió.

ERROR-2 0053H 83d

Call 0053H se diferencia de **RST 8H** sólo en que no actualiza la variable XPTR.

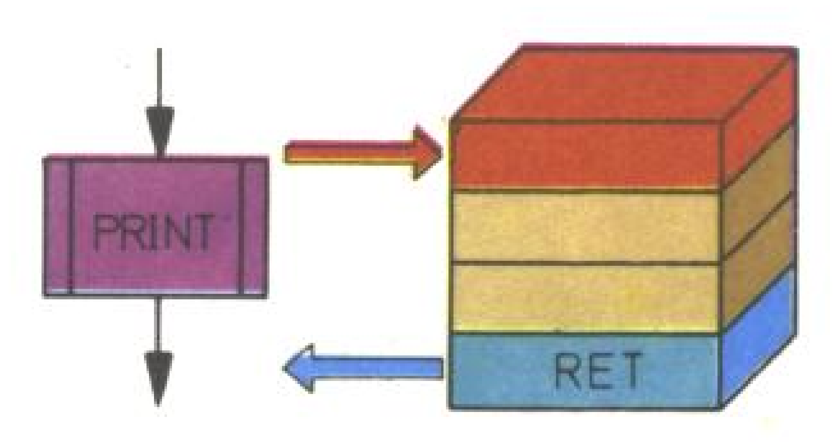
ERROR-3 0055H 85d

Esta rutina es como **ERROR-2** pero se llama con JP 0055H en lugar de CALL y el código de error menos 1 debe colocarse en el registro L en lugar de en el byte siguiente a la llamada.

Datos de entrada: L = código de error menos 1.
Datos de salida : SP = (ERRSP), HL = (STKEND).

Registros modificados: HL, SP.
Variables modificadas: ERR-NR, STKEND, MEM.

Rutinas que utiliza: SET-STK (16C5H). La rutina de error señalada indir. por (ERR-SP).
Rutina usada por : TEST-ROOM (1F05H).



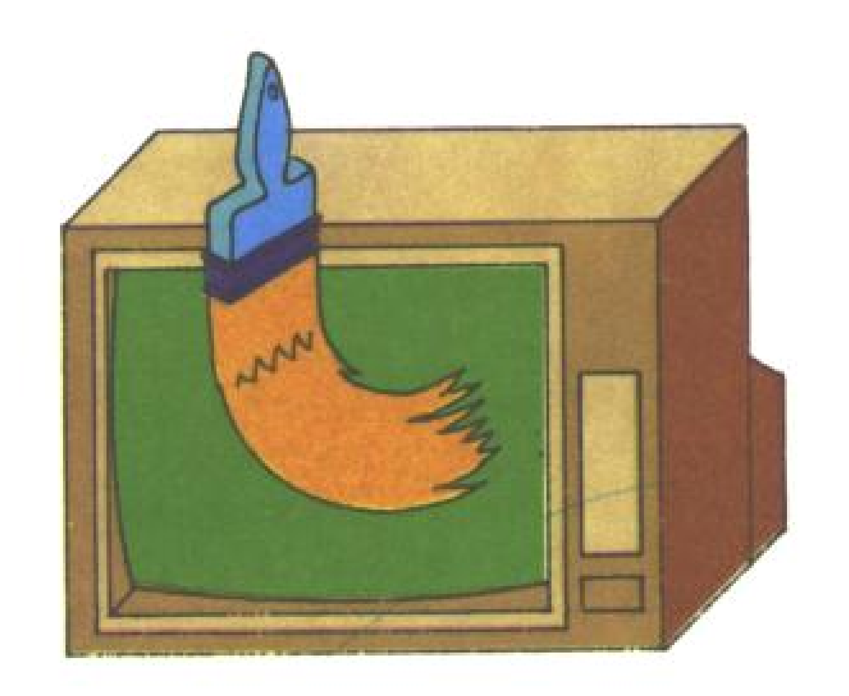
Nombre	Hex.	Dec.	
PRINT-A-1	0010H	16d	RST
GET-CHAR	0018H	24d	RST
TEST-CHAR	001CH	28d	
NEXT-CHAR	0020H	32d	RST

PRINT-A-1 0010H 16d

Rutina de presentación de un carácter: Utiliza la rutina **PRINT-A-2** situada en la dirección 15F2H que lee la dirección de la rutina correspondiente al canal de datos abierto en ese momento. Termina llamando a esa dirección.

Datos de entrada: A = Código del caracter.
Datos de salida : Según rutina correspondiente al canal.

Registros modificados: A, DE', BC'.
Variables modificadas: Las correspondientes al canal que se utilice.



Rutinas que utiliza: PRINT-A-2 (15F2H), CALL-SUB (15F7H). Rutina del canal abierto.
Rutina usada por : LOAD, LIST, PRINT, ETC.
Observaciones: Para usar RST 10H debe abrirse anteriormente el canal correspondiente, ej:

```
LD    A,2
CALL  1601H
```

abre el canal de la parte superior de la pantalla con lo que con RST 10 se podrá escribir en ella. (El canal 1 es la parte inferior de la pantalla y el 3 la impresora).

GET-CHAR 0018H 24d

Sitúa en el acumulador el caracter señalado por CH-ADD si éste es presentable en pantalla. Si se trata de un código de control lo salta así como sus parámetros correspondientes (1 para INK, etc, 2 para AT y TAB) devolviendo el próximo caracter presentable y actualizando (CH-ADD).

Datos de entrada: (CH-ADD) = Caracter actual.
Datos de salida : A = Caracter imprimible, (no de control).
— Flag Z alzado si el caracter es 0DH (ENTER).

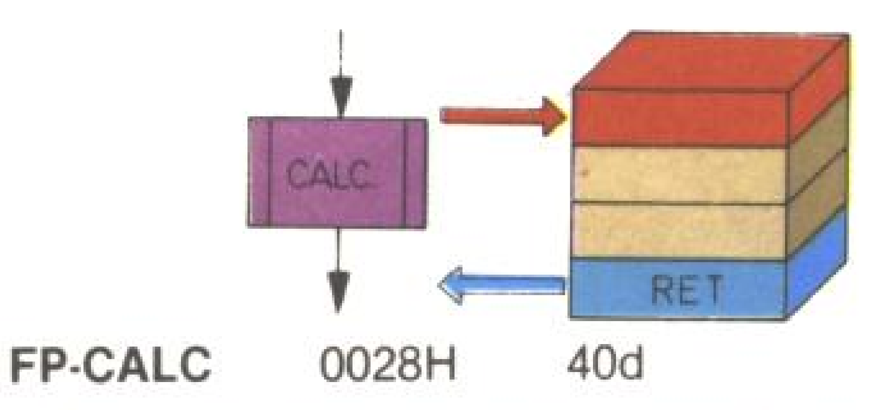
Registros modificados: A, HL.
Variables modificadas: CH-ADD.
Rutinas que utiliza: SKIP-OVER (007DH), NEXT-CHAR (0020H).
Rutina usada por : Múltiples rutinas.

NEXT-CHAR 001CH 28d

Hace lo mismo que RST 18H pero a partir del caracter siguiente.

Rutinas que utiliza: CH-ADD + 1(0074H), TEST-CHAR (001CH) continuación de GET-CHAR (0018H).
Observaciones: El resto de los datos como GET-CHAR (RST 18H).





FP-CALC 0028H 40d

Rutina del calculador en coma flotante. Inmediatamente después de la llamada a esta rutina deben estar los códigos de las operaciones que se deseen realizar terminados por el código 38H END-CALC (Fin de los cálculos). La rutina termina retornando a la dirección siguiente de donde se encuentre el código 38H.

Datos de entrada: Tabla con las operaciones a realizar inmediatamente después de la llamada a la rutina.

Datos de salida : En el stack del calculador.

Registros modificados: Múltiples

Variables modificadas: BREG, STKEND, etc.

Nombre	Hex.	Dec.	
FP-CALC	0028H	40d	RST
BC-SPACES	0030H	48d	RST

Rutinas que utiliza: CALCULATE (335BH).

Rutina usada por : Múltiples comandos.

Observaciones: Los datos previos han de introducirse en el stack del calculador con alguna de las siguientes rutinas:



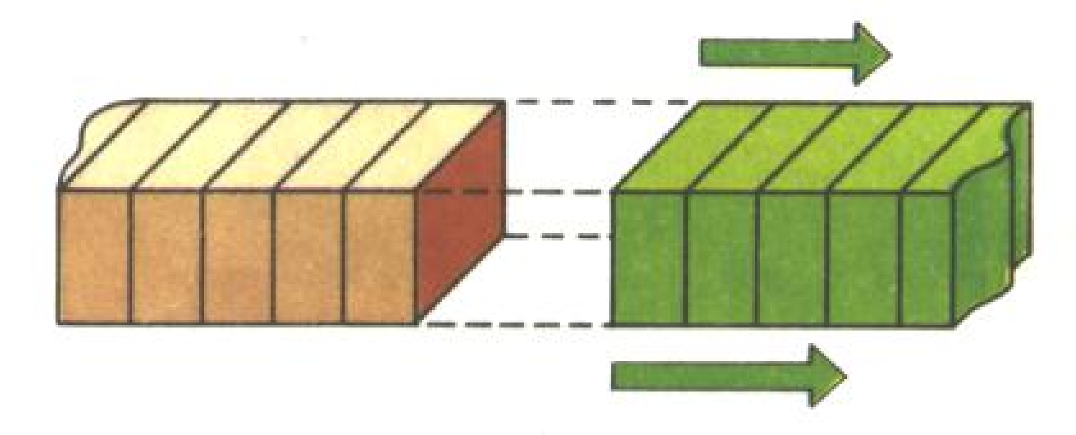
STACK-A	2D28H	11560d	0< =n= >255
STACK-BC	2D2BH	11563d	—65535< =n = >65535
STK-ST-0	2AB1H	10929d	coma flotante
STK-NVM	33B4H	13236d	coma flotante
SLICING	2A52H	10834d	cadena alfanum.
INT-TO-FP	2D3BH	11579d	cadena núm. ent.
DEC-TO-FP	2D9BH	11675d	cadena núm.

Para extraer datos del calculador se pueden utilizar las siguientes rutinas:

FIND-INT1	1E94H	7828d	0< =n= >255
FP-TO-A	2DD5H	11733d	
FIND-INT2	1E99H	7833d	—65535< =n
PF-TO-BC	2DA2H	11682d	= <65535

BC-SPACES 0030H 48d

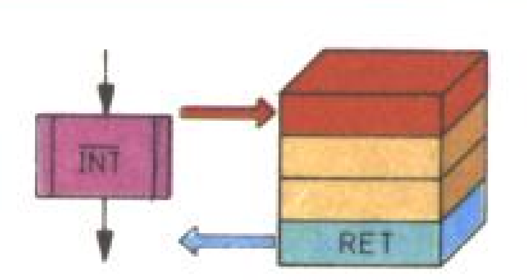
Crea una zona libre en el espacio de trabajo (Work space) de una longitud determinada por el par de registros BC. Está lugar se hace entre el espacio de trabajo anterior y el stack del calculador.



Datos de entrada: BC: Número de bytes.
Datos de salida : DE: Primer byte extra.
HL: Ultimo byte extra.
BC: Como entró.

Registros modificados: DE,HL,BC.
Variables modificadas: WORK-SP, STK-BOT y STK-END.

Rutinas que utiliza: RESERVE (169EH) y MAKE-ROOM (1655H).
Rutina usada por : Diversas rutinas.
Observaciones: Para eliminar todos los espacios de trabajo puede utilizarse la rutina SET-MIN (16B0H).



MASK-INT 0038H 56d

Rutina llamada por las interrupciones enmascarables ($\overline{INT}$) en el modo 1 de interrupciones (IM1) 50 veces por segundo.

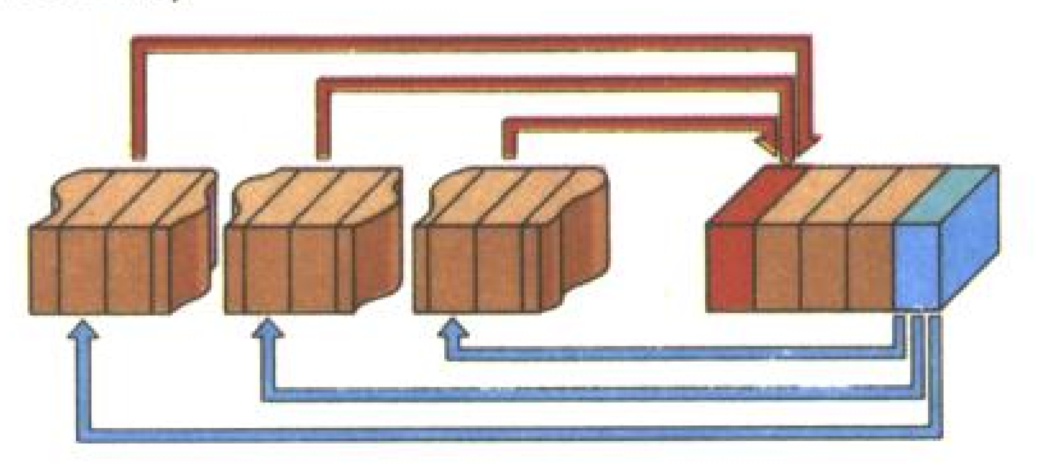
Incrementa en una unidad el contador FRAMES e inspecciona el teclado.

Datos de entrada: Ninguno.
Datos de salida : Ninguno.

Registros modificados: Ninguno.
Variables modificadas: FRAMES y las relativas a la inspección del teclado: KSTATE, FLAGS y LASTK.
Rutinas que utiliza: KEYBOARD (02BFH).
Rutina usada por : El modo 1 de interrupciones enmascarables.

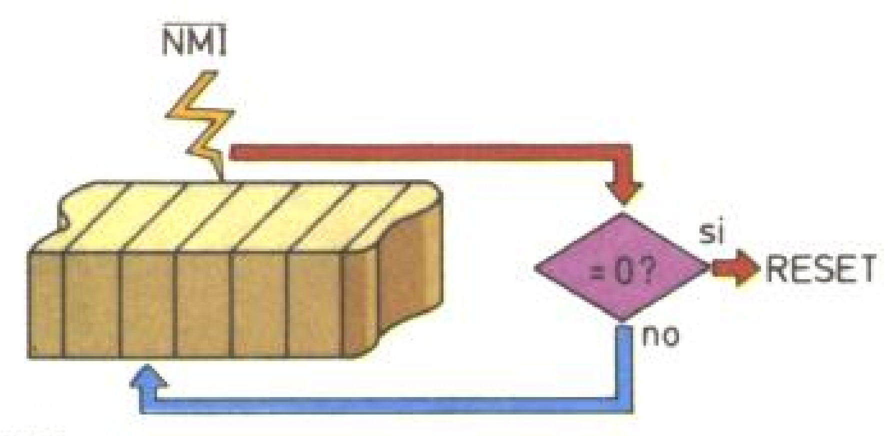
Nombre	Hex.	Dec.		
MASK-INT	0038H	56d	RST	$\overline{INT}$
ERROR-2	0053H	83d		
ERROR-3	0055H	85d		
RESET	0066H	102d		$\overline{NMI}$

Observaciones: Cuando se use otro modo de interrupción (ej: IM2) o estén deshabilitadas las interrupciones DI, deberá hacerse RST 38H (RST 56 dec.) para poder atender al teclado o, en su defecto, alguna rutina que lo atienda propia del programador o la de la ROM **(KEYBOARD)**.



ERROR-2 ERROR-3

Ver microficha M-1.



RESET

Rutina de interrupciones no enmascarables: Es llamada por hardware al ser activada la patilla $\overline{NMI}$ del microprocesador.

Produce un Reset: rutina START (CALL 0) si la variable del sistema NMIADD (5CB0H = 23728d) es 0. No produce ningún efecto si contiene cualquier otro valor.

Datos de entrada: Ninguno.
Datos de salida : Ninguno.
Registros modificados: Ninguno (o todos si ejecuta START).
Variables modificadas: Ninguna (o todas si ejecuta START).
Rutinas que utiliza: Ninguna o START (0).
Rutina usada por : Las interrupciones no enmascarables.

Observaciones: Esta rutina así como está no es muy útil. Los señores de Sinclair se equivocaron al hacer la ROM y pusieron JR NZ donde debiera ser JR Z. Si hubiese sido así la rutina terminaría con un salto a la dirección señalada por NMIADD y retornaría en caso de que esta variable contuviese un 0. De esta forma podríamos ejecutar cualquier rutina por hardware.

De todas formas este error puede suplirse en cierta manera haciendo que una rutina ejecutable en el modo 2 de interrupciones enmascarables consulte un determinado port y si está activado hacer un salto a la dirección que se desee. En este caso el dispositivo externo debería estar conectado a ese port y no a $\overline{NMI}$.

CH-ADD + 1 0074H 116d

Incrementa en 1 el valor de la variable CH-ADD y sitúa en A el byte que señala.

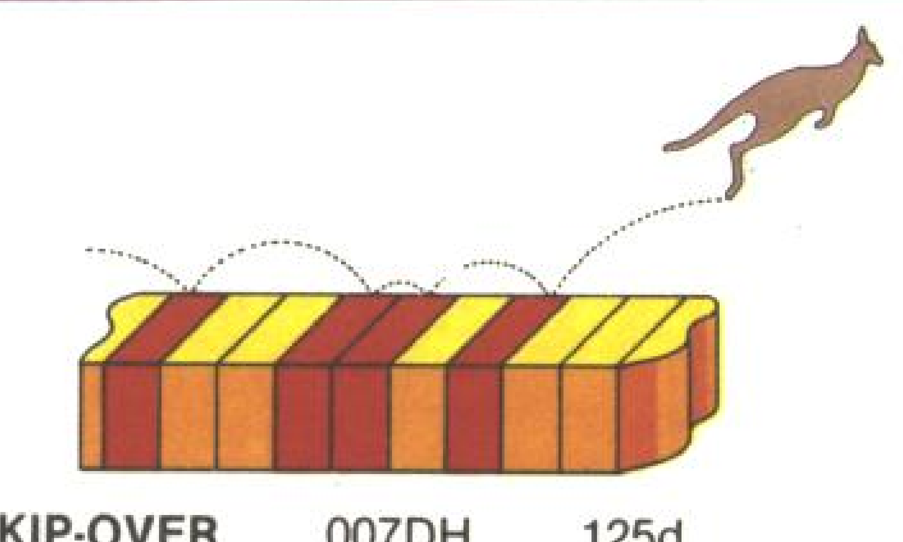
Datos de entrada: Ninguno.
Datos de salida : CH-ADD incrementado en 1.
HL = cont. de (CH-ADD).
A = Carácter señalado.

Registros modificados: A, HL.
Variables modificadas: CH-ADD.

Rutinas que utiliza: Ninguna.
Rutina usada por : **SCANNING** (24FBH), **INT-TO-FP** (2D3BH).

Observaciones: Esta rutina tiene otras dos posibles entradas: **TEMP-PTR1** (0077H) y **TEMP-PTR2** (0078H) que son usadas para modificaciones temporales de CH-ADD por la rutina del comando **READ** (1DEDH).

Nombre	Hex.	Dec.	
CH-ADD + 1	0074H	116d	
TEMP-PTR-1	0077H	119d	
TEMP-PTR-2	0078H	120d	
SKIP-OVER	007DH	125d	
TOKEN-TABLE	0095H	149d	Tab. de inst.
KEY-TABLES	0205H	517d	Tab. de tec.



SKIP-OVER 007DH 125d

Comprueba el valor de A e incrementa el valor de CH-ADD 1 ó 2 unidades si éste es un código de control con parámetros.

Datos de entrada: HL: Dirección del caracter por comprobar.
A: Código del caracter.

Datos de salida : CH-ADD actualizado.
HL actualizado.
Carry: Si A > 20H.
Flag Z si A = 0DH (ENTER).

Registros modificados: HL.
Variables modificadas: CH-ADD.

Rutinas que utiliza: Ninguna.
Rutina usada por : GET-CHAR (0018H) y NEXT-CHAR (0020H).

TOKEN-TABLE 0095H 149d

Todas las instrucciones del Spectrum están enumeradas en esta tabla. Su finalidad es ser escritas a partir de un solo byte. Para reconocer el último caracter de cada palabra éste está invertido (bit 7 puesto a 1).

KEY-TABLE 0205H 517d

Tablas de las teclas; se utiliza para establecer la correspondencia entre la posición de cada una y el código de caracter con que se corresponde según el modo en que se encuentre.

0205H (517d): Tabla de las teclas en modo L + CAPS SHIFT (Números, letras, ENTER, SYMBOL y SPACE).
022CH (557d): Tabla de las funciones en modo E (READ, BIN, etc.)
0246H (582d): Tabla de las funciones y gráficos en modo E; Teclas de letras + SYMBOL SHIFT. (BRIGHT, etc.).
0260H (608d): Tabla de los códigos de control: Teclas numéricas + CAPS SHIFT (DELETE, EDIT, etc.).
026AH (618d): Tabla de los comandos y gráficos en modo L + SYMBOL SHIFT (STOP, *, etc.).
0284H (644d): Tabla de los comandos en modo E; Teclas numéricas + SYMBOL SHIFT (FORMAT, DEF FN, etc.).

KEY-SCAN 028EH 654d

Rutina de exploración del teclado. Lee todos los puertos del teclado devolviendo en el registro E cuál es la tecla que está siendo pulsada. Las teclas están numeradas de 0 a 39 (27H) siguiendo una espiral en el teclado.

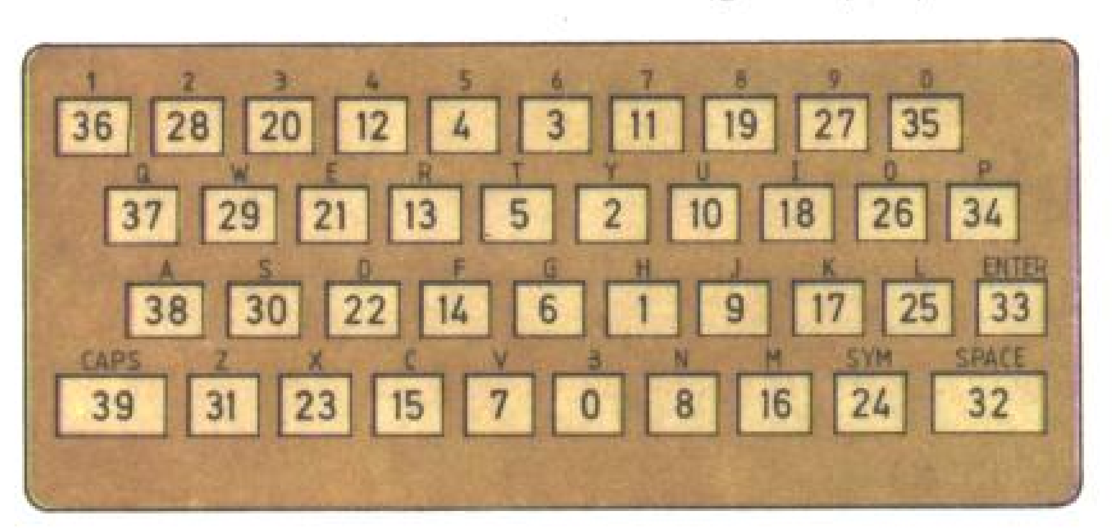
El flag indicador de cero (Z) sirve para indicar si la combinación de teclas pulsada es correcta o no.

Datos de entrada: Ninguno.

Datos de salida :

- Ninguna tecla pulsada:
  E = FFH, D = FFH.
  Zero Flag = 1 (Z).
- Una tecla pulsada:
  E = Núm. Tecla.
  D = FF, Zero Flag = 1 (Z).
- Tecla + CAPS o SYM:
  E = Núm. Tecla.
  D = 27H (CAPS) o 18H (SYM).
  Zero Flag = 1 (Z).
- Pulsadas CAPS y SYM.
  E = 27H (CAPS), D = 18H (SYM.).
  Zero Flag = 1 (Z).
- 2 tec. (CAPS ni SYM.).
  E = núm. de tec. mayor.
  D = núm. de tec. menor.
  Zero Flag = 0 (NZ).

Nombre	**Hex.**	**Dec.**
KEY-SCAN	028EH	654d
KEYBOARD	02BFH	703d



- Más de 2 teclas pulsadas:
  D y E desconocidos.
  Zero Flag = 0 (NZ).

Registros modificados: A, BC, HL, DE.
Variables modificadas: Ninguna.
Rutinas que utiliza: Ninguna.
Rutina usada por : **KEYBOARD** (02BFH).
S-INKEI$ (2634H).

Observaciones: Cuando más de dos teclas han sido pulsadas, los valores de D y E suelen coincidir con los que resultan de pulsar otras dos teclas diferentes, por lo que no es segura la rutina para la comprobación de la pulsación de dos teclas concretas.

KEYBOARD 02BFH 703d

Rutina de consulta del teclado llamada cada 20 milisegundos por las interrupciones enmascarables MASK INT (RST 38). Su misión es colocar el código de la tecla pulsada en la variable LAST-K.

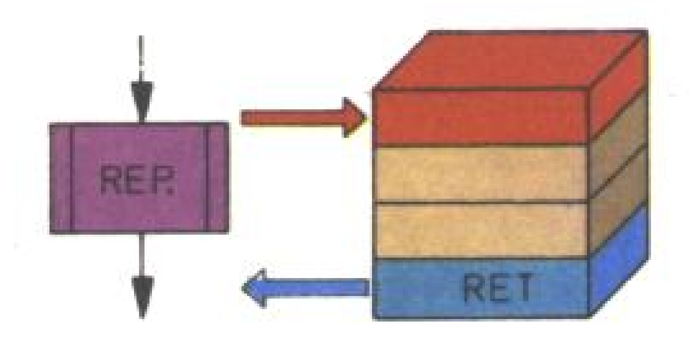
Debe tener en cuenta las variables de retardo REPDEL y REPPER para repetición de teclas.

Para contabilizar estos períodos utiliza el doble sistema de variables (KSTATE0-KSTATE3 y KSTATE4-KSTATE7).

Datos de entrada: REPPER, REPDEL.
Datos de salida : HL = KSTATE3 o KSTATE7
A y (LAST-K). Ultima tecla pulsada, sólo si lo permitieron REPDEL y REPDEL.
SET 5, (FLAGS). En el caso anterior.

Registros modificados: A, BC, DE, HL.
Variables modificadas: KSTATE0-KSTATE7, FLAGS, LASTK.
Rutinas que utiliza: **KEY-SCAN** (028EH).
K-REPEAT (0310H).
K-TEST (031EH).
K-DECODE (0333H).
Rutina usada por : **MASK-INT** (0038H); Interrupciones enmascarables.





Nombre	Hex.	Dec.
K-REPEAT	0310H	784d
K-TEST	031EH	798d
K-DECODE	0333H	819d

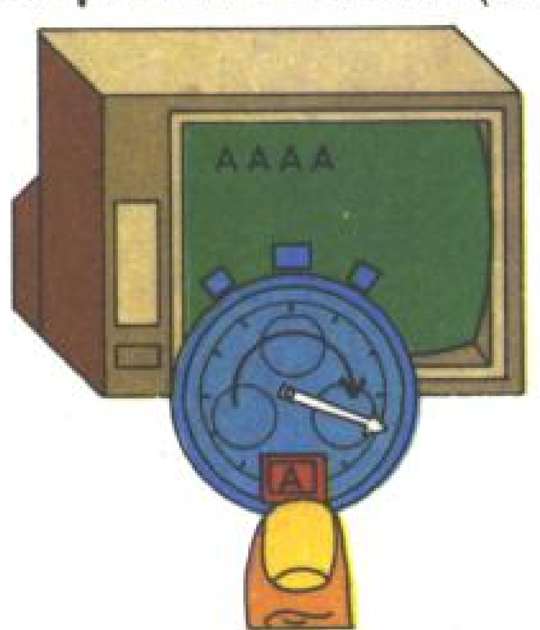
K-REPEAT 0310H 784d

Esta rutina es llamada por KEYBOARD cuando se mantiene pulsada la misma tecla. Su misión es decrementar el contador de retardo y sólo si éste llega a 0, aceptar la repetición de tecla. En este caso es inicializado el contador con el valor de REPPER (normalmente 0.1 seg.). La primera vez el valor del retardo viene dado por REPDEL (normalmente 0.7 seg.).

Datos de entrada: HL = KSTATED/4, REPPER, KSTATE.

Datos de salida : Ninguno si no es tiempo. (LAST-K) = A y SET 5, (FLAGS) si se cumplió el retardo.

Registros modificados: A, HL.
Variables modificadas: KSTATE.
Rutinas que utiliza Ninguna.
Rutina usada por: **KEYBOARD** (02BFH).



K-TEST 031EH 798d

Esta rutina retorna con el Flag NZ si no hay tecla pulsada, o si sólo ha sido pulsada una de entre CAPS o SYMBOL SHIFT.

En caso contrario, es activado el Flag Z y devuelto en el acumulador el código de la letra en modo C según la tabla principal de teclas situada en la dirección 0202H.

Datos de entrada: D y E como salieron de **KEY-SCAN** (028EH).

Datos de salida : B = anterior D.
D = 0, E como entró.

- Si pulsación incorrecta:
  A = E.
  Carry Flag = 0 (NC).
- Si pulsación correcta:
  A Cód. carac. modo «C».
  HL Dir. cód. en K-MAIN.
  Carry Flag = 1 (C).

Registros modificados: A, B, D, HL.
Variables modificadas: Ninguna.
Rutinas que utiliza: Ninguna.
Rutina usada por : **KEYBOARD** (02BFH).

K-DECODE

Decodificador de teclado. A partir del código principal calculado por K-TEST y guardado posteriormente en el registro E esta rutina calcula el código real.

Datos de entrada: E = código principal.
D = (FLAGS), C = (MODE).
B = Valor de SHIFT.
Datos de salida : A = Código del caracter.

Registros modificados: A, BC, D, HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **KEYBOARD** (02BFH).
S-INKEY$ (2634H).

BEEPER 03B5H 949d

El sonido del Spectrum es producido por la activación y desactivación intermitente (frecuencia) del bit 4 del port «254» (FEH) durante un tiempo determinado. Este tiempo ha de estar expresado en T estados de reloj. (1 seg. = 66894d estados).

Datos de entrada: DE = Frecuencia ✱ tiempo.
HL = T estados/4—30 =
= Tiempo en seg:
✱ 6689/ 4—30.

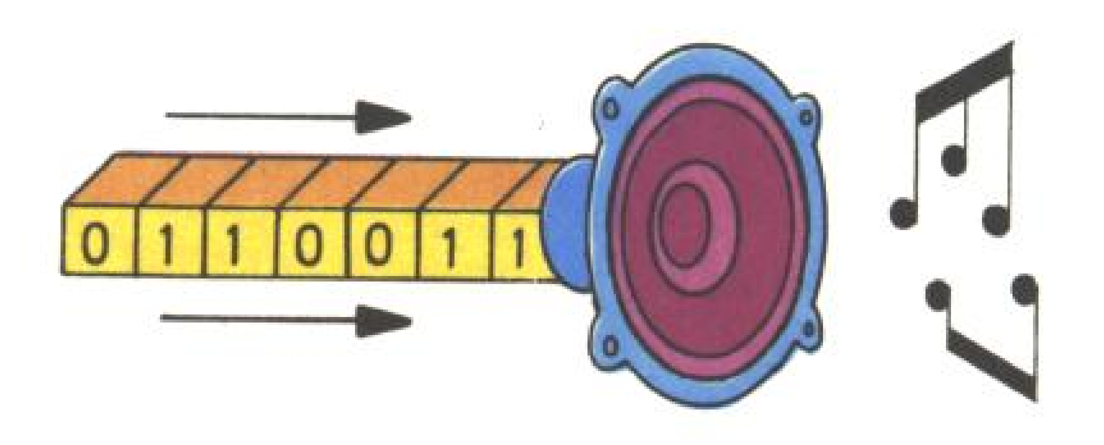
Datos de salida : Ninguno.

Registros modificados: A, BC, DE, HL, IX.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **BEEP** (03F8H).
ED-LOOP (0F38H).
ED-ERROR (10F7H).
ED-FULL (1167H).

Nombre	Hex.	Dec.	
BEEPER	03B5H	949d	
BEEP	03F8H	1016d	COMANDO
S-TONE-T	046EH	1134d	TABLA

Observaciones: Esta rutina deshabilita las interrupciones enmascarables durante su ejecución, habilitándolas al terminar. Por esta razón la variable FRAMES, usada como contador de tiempo, no será incrementada.



BEEP 03F8H 1016d

Rutina del comando **BEEP.** Efectúa los cálculos de los datos necesarios como entrada en la rutina **BEEPER.**

Datos de entrada: El tiempo y la nota deben encontrarse en el stack del calculador (STK).
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: Las del STK del calc.

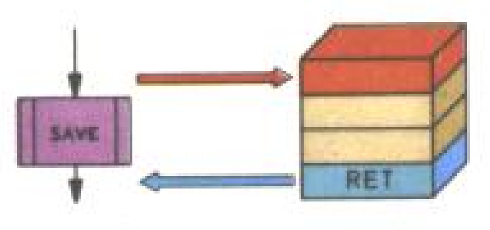
Rutinas que utiliza: **FP-CALC** (0028H) RST.
BEEPER (03B5H).
LOC-MEM (3406H).
STACK-NUM (33B4H).
FIND-INT1 (1E94H).
FIND-INT2 (1E99H).
Rutina usada por : El comando BEEP del BASIC.

SEMI-TONE TABLE

Tabla de semitonos. Es utilizada por **BEEP** para obtener la frecuencia de la nota correspondiente:

Frecuencia hz.	note	nota
261,63	C	DO
277,18	C#	DO#
293,66	D	RE
311,13	D#	RE#
329,63	E	MI
349,23	F	FA
369,99	F#	FA#
392,00	G	SOL
415,30	G#	SOL#
440,00	A	LA
466,16	A#	LA#
493,88	B	SI





SA-BYTES 04C2H 1218d

Salva en cassette un bloque de bytes. Es llamada dos veces, una para salvar la cabecera y otra para salvar el programa o bloque de datos.

Puede usarse por el programador para salvar programas sin cabecera.

Datos de entrada: DE = Longitud del bloque.
IX = Comienzo del bloque.
A = Código de control:
00H Cabecera.
FFH Programa o datos.

Datos de salida : IX = Final del bloque + 2.
DE = FFFF H.

Registros modificados: AF,BC,DE,HL,IX,AF'.
Variables modificadas: Ninguna.
Rutinas que utiliza: **SA/LD-RET** (053FH).
Rutina usada por : **SA-CONTRL** (0970H).

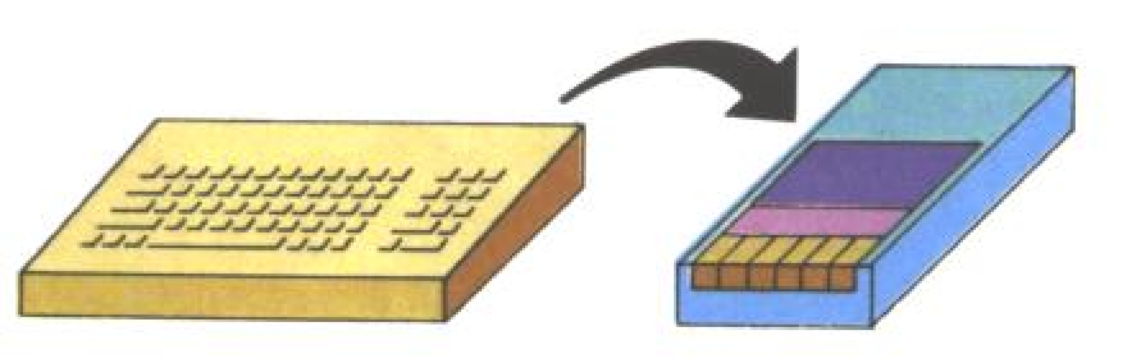
Nombre	Hex.	Dec.	
SA-BYTES	04C2H	1218d	SAVE
SA/LD-RET	053FH	1343d	
LD-BYTES	0556H	1366d	LOAD
LD-EDGE2	05E3H	1507d	
LD EDGE1	05E7H	1511d	
SAVE-ETC	0605H	1541d	ENTRADA
VR-CONTRL	07CBH	1995d	COMANDOS
LD-BLOCK	0802H	2050d	
LD-CONTRL	0808H	2056d	
ME-CONTRL	08B6H	2230d	
ME-ENTER	092CH	2348d	
SA-CONTRL	0970H	2416d	

Observaciones: El código de control que debe entrar en el Acumulador puede ser cualquier otro número, que será necesario para volver a cargar el bloque. De este modo, puede usarse como clave.

Esta rutina durante su funcionamiento deshabilita las interrupciones.

SA/LD-RET 053FH 1343d

Es la salida común de las rutinas de salvar y cargar. Restablece el BORDER original y habilita las interrupciones.



SAVE-ETC 0605H 1541d

Esta es la entrada común de los cuatro comandos **SAVE,LOAD,VERIFY** y **MERGE.** Su misión es construir la nueva cabecera en el espacio de trabajo, leer la antigua cabecera de cassette, si es necesario, escribiendo los mensajes en pantalla y comparar los nombres. Por último salta a la rutina de control correspondiente al comando.

SA-CONTRL 0970H 2416d

Rutina de grabación de programa o datos con cabecera.

Datos de entrada: HL = Dirección del bloque.
IX = Dirección de la cabecera.

Datos de salida : IX = Final del bloque + 2.
DE = FFFFH.

Registros modificados: AF,BC,DE,HL,IX,AF'.
Variables modificadas: Relativas al canal K.
Rutinas que utiliza: **CHAN-OPEN** (1601H).
PO-MSG (0C0AH).
WAIT-KEY (15D4H).
SA-BYTES (04C2H).
Rutina usada por : **SAVE-ETC** (0605H).

Observaciones: Si no se desea que se imprima el mensaje ni espere la pulsación de una tecla, ha de hacerse:

```
PUSH   HL
CALL   0984H ;   2436d
```

LD-BYTES 0556H 1366d

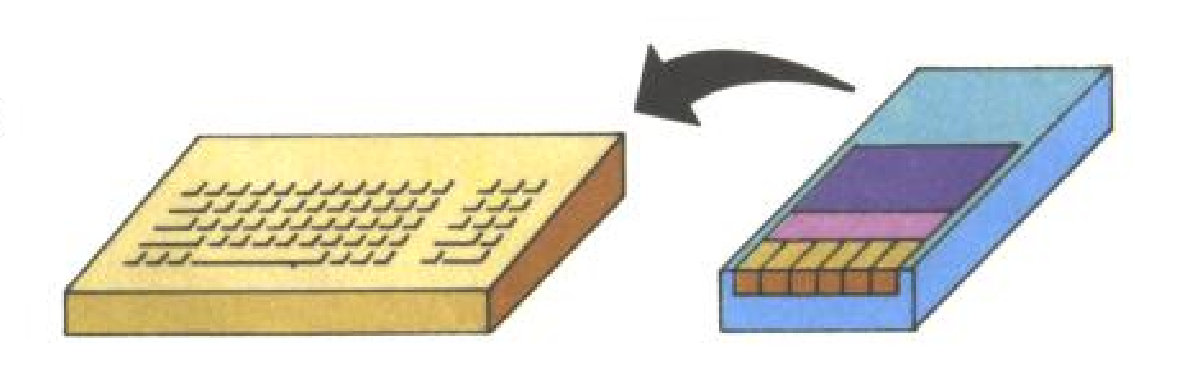
Carga o verifica un bloque de bytes del cassette. Es llamada dos veces, una para cargar la cabecera y otra para cargar o verificar un programa o bloque de datos.

Puede usarse por el programador para cargar o verificar programas sin cabecera.

Datos de entrada: DE = Longitud del bloque.
IX = Comienzo del bloque.
A = Código de control:
00 Cabecera.
FF Programa o datos.
Carry = 1 (C) : LOAD.
= 0 (NC): VERIFY

Datos de salida : IX = Ultimo byte cargado correctamente + 1.
— Si carga correcta:
DE = 0, Carry flag (C).
— Si carga incorrecta:
Carry flag = 0 (NC)
— Si código incorrecto:
L = Código.

Nombre	Hex.	Dec.	
LD-BYTES	0556H	1366d	LOAD
LD-EDGE2	05E3H	1507d	
LD-EDGE1	05E7H	1511d	
SAVE-ETC	0605H	1541d	
VR-CONTRL	07CBH	1995d	
LD-BLOCK	0802H	2050d	
LD-CONTRL	0808H	2056d	



Registros modificados: AF,BC,DE,HL,IX,AF'.
Variables modificadas: Ninguna, salvo si son cargadas directamente.
Rutinas que utiliza: **LD-EDGE2/1** (05E3H/05E7H). **SA/LD-RET (053FH).**
Rutina usada por : **LD-BLOCK (0802H). (LOAD,VERIFY,MERGE).**

Observaciones: El código de control que debe entrar en el Acumulador debe ser el mismo que aquél con que el bloque fue salvado (Normalmente 0 para cabecera y FFH para bloque de datos). En caso contrario el bloque no se cargará pero se cargará su código en el registro L.

Esta rutina durante su funcionamiento deshabilita las interrupciones.

LD EDGE2/1 05E3H/05E7H 1507d/1511d

Estas subrutinas son la parte más importante de LOAD y VERIFY. Comprueban los cambios de señal en la entrada de cassette (port 7FFEH) que determinarán si los bits que entran son ceros o unos; cambian el color del BORDER y detectan si fue pulsado BREAK.

SAVE-ETC **VR-CONTRL**

Ver microficha M-9.

LD-BLOCK 0802H 2050d

Llama a LD-BYTES y produce un mensaje de error si la carga o verificación es incorrecta. Es usada por LOAD y VERIFY.

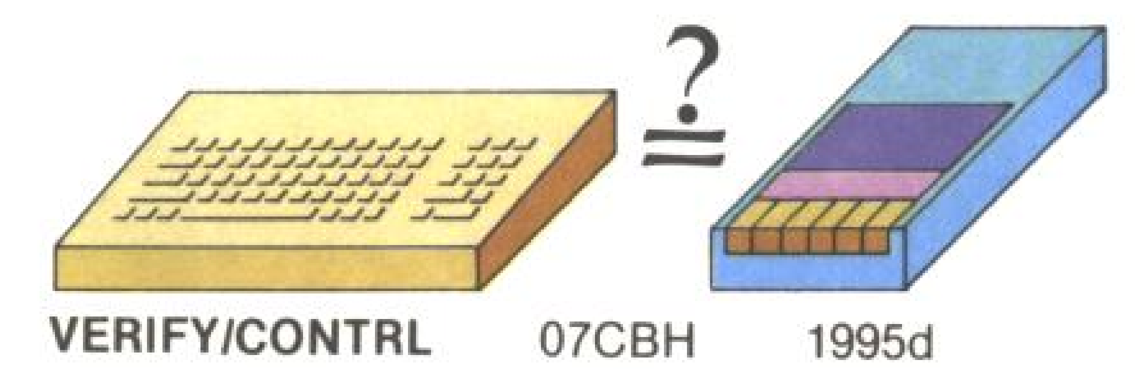
Puede usarse en lugar de LD-BYTES para cargar o verificar programas sin cabecera.

LD-CONTRL 0808H 2056d

Rutina de control de carga de un programa BASIC y sus variables o un «array» (variable dimensionada).

Comprueba si hay sitio para lo que va a cargar, moviendo la memoria si es necesario. Ajusta las variables del sistema al nuevo programa y termina saltando a LD-BYTES.





VERIFY/CONTRL 07CBH 1995d

Esta rutina es usada por todos los casos de VERIFY y para LOAD «SCREENS» o «CODE».

Comprueba la longitud del programa que va a entrar. Si es correcta, entra en la rutina **LD-BLOCK** para verificar un programa o datos, o para cargar datos.

LD-BLOCK **LD-CONTRL** Ver microficha M-10.

ME-CONTRL 08B6H 2230d

Control de unión de programas. Se realiza en tres partes:

a) Carga el bloque de datos en el espacio de trabajo.

b) Cambia o añade nuevas líneas al programa antiguo.

c) Cambia o añade nuevas variables.

Nombre	Hex.	Dec.	
VR-CONTRL	07CBH	1995d	
LD-BLOCK	0802H	2050d	
LD-CONTRL	0808H	2056d	
ME-CONTRL	08B6H	2230d	
ME-ENTER	092CH	2348d	
SA-CONTRL	0970H	2416d	
CASS-MES	09A1H	2465d	TABLA

Datos de entrada: IX = Dirección de la cabecera.

Datos de salida : H L = Fin del nuevo programa.

Registros modificados: AF,BC,DE,HL,IX,AF'.

Variables modificadas: Punteros del BASIC.

Rutinas que utiliza: **BC-SPACES** (0030H). **ME-ENTER** (092CH).

Rutina usada por : **SAVE-ETC** (0605H).

Observaciones: Para hacer Merge de un programa sin cabecera debe cargarse en BC la longitud y llamar a rutina en la dirección 08BCH (2236d).

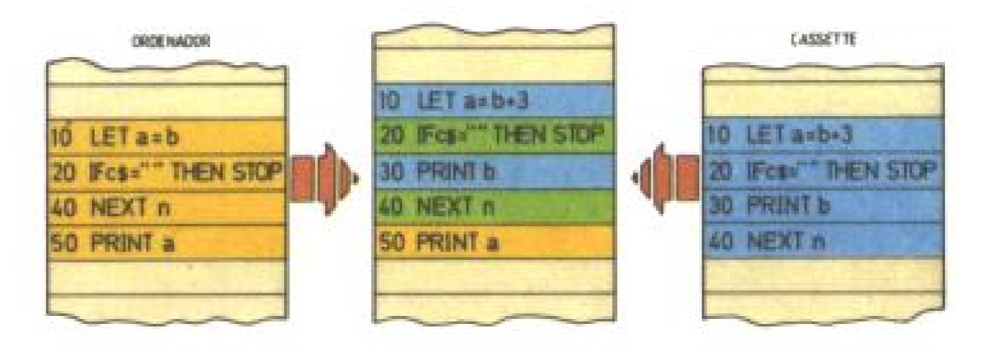
ME-ENTER 092CH 2348d

Une o sustituye una línea o variable del programa cargado, en el antiguo.

Datos de entrada: HL = Dirección de la nueva línea o variable.
DE = Lugar donde debe colocarse.
Carry = 1 (C) = Variable.
= 0 (NC) = Línea BASIC.
Flag Z = 1 (Z) = Sustit.
= 0 (NZ) = Unión.

Datos de salida : HL = Comienzo siguiente línea o variable en nuevo programa.
DE = Idem en el antiguo.

Registros modificados: AF,BC,DE,HL,AF'.
Variables modificadas: Punteros del BASIC.
Rutinas que utiliza: **NEXT-ONE** (19B8H).
RECLAIM-2 (19E8H).
MAKE-ROOM (1655H).
Rutina usada por : **ME-CONTRL** (08B6H).



SA-CONTRL Ver microficha M-9.

CASS-MES 09A1H 2465d

Cada mensaje termina con un carácter invertido (bit 7 = 1). El carácter anterior a un mensaje también debe tener alzado el bit 7.

Para presentar un mensaje se utliza la rutina PO-MSG (0C0AH). Debe encontrarse en DE una dirección anterior al mensaje, y en A el lugar que ocupa ese mensaje a partir de esa dirección.

09A1	Carácter de comienzo de mensaje (80H).	
09A2	Start tape, then press any key.	
09C1	ENTER	Program:
09CB	ENTER	Number array:
09DA	ENTER	Character array:
09EC	ENTER	Bytes:

PRINT-OUT 09F4H 2548d

Rutina de salida de datos de los canales:
1-K-Parte inferior de la pantalla.
2-S-Parte superior de la pantalla.
3-P-Impresora.

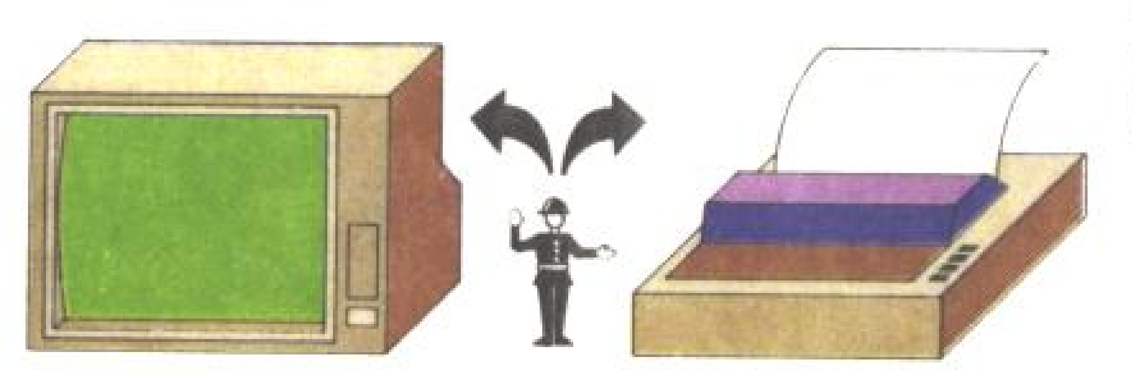
La rutina RST 10H lee en CURCHL esta dirección cuando ha sido abierto alguno de estos canales con la rutina CHAN-OPEN (1601H).

Esta rutina concluye con un salto a:

PO-QUEST si es un caracter del 0 al 5 (no usados) para imprimir un signo de interrogación.

La rutina señalada por la tabla CONT-CHAR si es un caracter de control.

PO-ABLE si es un caracter ordinario, gráfico o TOKEN.



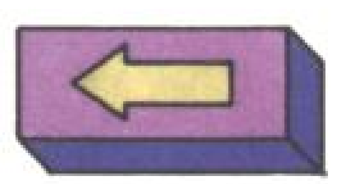
Nombre	Hex.	Dec.	
PRINT-OUT	09F4H	2548d	PRINT
CONT-CHAR	0A11H	2577d	TABLA
PO-BACK1	0A23H	2595d	
PO-RIGHT	0A3DH	2521d	
PO-ENTER	0A4FH	2639d	
PO-COMMA	0A5FH	2655d	Carácteres
PO-QUEST	0A69H	2665d	de
PO-TV-2	0A6DH	2669d	control
PO-CHANGE	0A80H	2688d	
PO-CONT	0A87H	2695d	
PO-ABLE	0AD9H	2777d	

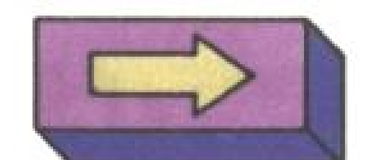
Datos de entrada: A = Código del caracter.
Datos de salida : Ninguno.

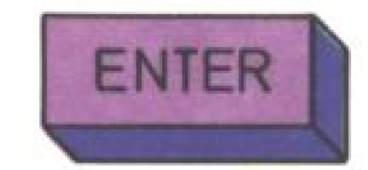
Registros modificados: Múltiples.
Variables modificadas: Las relativas al canal utilizado.
CURCHL si se trata de un caracter de control con parámetros.

Rutinas que utiliza: **PO-FETCH** (0B03H).
PO-ABLE (0AD9H).
PO-QUEST (0A69H).
Rutinas de los caracteres de control.

Rutina usada por : **PRINT-A-2** (15F2H) RST 10H.







CONT-CHAR 0A11H 2577d

Tabla de saltos de las rutinas de los caracteres de control (códigos 6 a 17H).

PO-BACK-1 0A23H 2595d

Cursor a la izquierda.

PO-RIGHT 0A3DH 2521d

Cursor a la derecha. Debido a un error esta rutina no termina saltando a **PO-STORE.**

PO-ENTER 0A4FH 2639d

Rutina de retorno de carro.

PO-COMMA 0A5FH 2655d

Dibuja espacios hasta completar media línea.

PO-QUEST 0A69H 2665d

Dibuja un signo de interrogación, para los caracteres no usados, mediante la rutina PO-ABLE.

Caracteres de control con operandos:

El código de control es salvado en el primer BYTE de la variable TVDATA y es cambiado el valor de CURCHL para que la próxima entrada no sea interpretada como un caracter, sino como uno o dos parámetros.

PO-ABLE 0AD9H 2777d

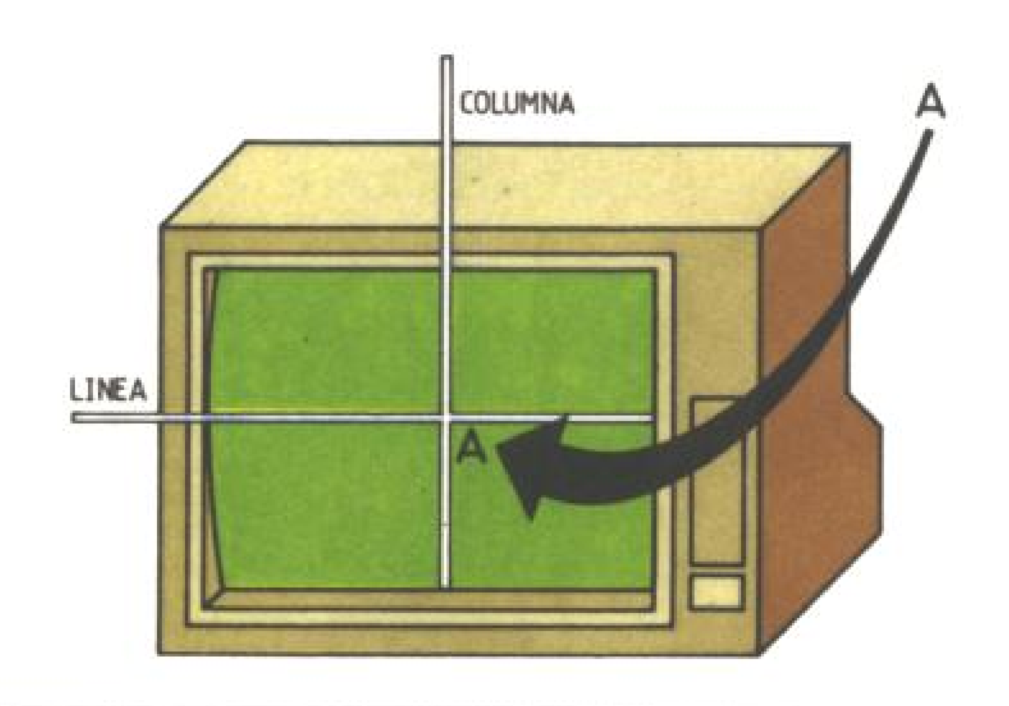
Llama a PO-ANY para presentar un caracter y entra en PO-STORE para actualizar la posición del cursor.

PO-STORE 0ADCH 2780d

Actualiza las variables de posición del cursor en el canal que se está utilizando.

Datos de entrada: BC Línea y columna invertidas.
HL Dirección de esa posición.

Datos de salida : Los mismos.



Nombre	Hex.	Dec.
PO-STORE	0ADCH	2780d
PO-FETCH	0B03H	2819d
PO-ANY	0B24H	2852d
PO-GR-1	0B38H	2872d
PO-T&UDG	0B52H	2898d

Registros modificados: Ninguno.
Variables modificadas: SPOSN y DF-CC o S-POSNL, ECHO-E y DF-CCL o P-POSN y PR-CC.

Rutinas que utiliza: Ninguna.
Rutina usada por : Las rutinas de presentación.

PO-FETCH 0B03H 2819d

Carga los parámetros de posición del canal en curso.

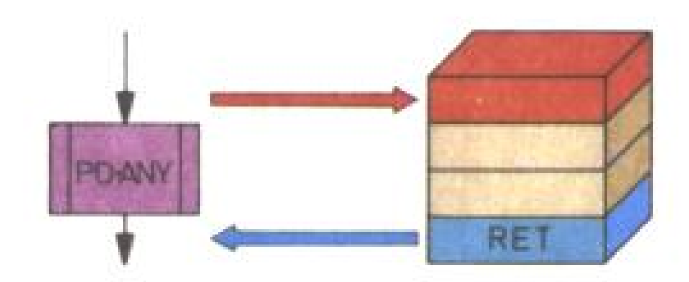
Datos de entrada: Bit 1 (FLAGS) y
Bit 0 (TV-FLAG).
Datos de salida : BC Línea y col. inversas.
HL Direc. de esa posición.

Registros modificados: BC, HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : Rutinas de presentación.

PO-ANY 0B24H 2852d

Imprime cualquier caracter que no sea de control saltando a la rutina correspondiente:
Caracter ordinario: **PO-CHAR.**
Gráfico ordinario: **PO-GR-1.**
Gráfico definido o TOKEN: **PO-T&UDG.**



PO-GR-1 0B38H 2872d

Construye un símbolo gráfico (códigos 128-143d) en MEMBOT.

Datos de entrada: B = Código del gráfico.
Datos de salida : Ninguno.

Registros modificados: AF,BC,HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **PO-ANY** (0B24H).

PO-T&UDG 0B52H 2898d

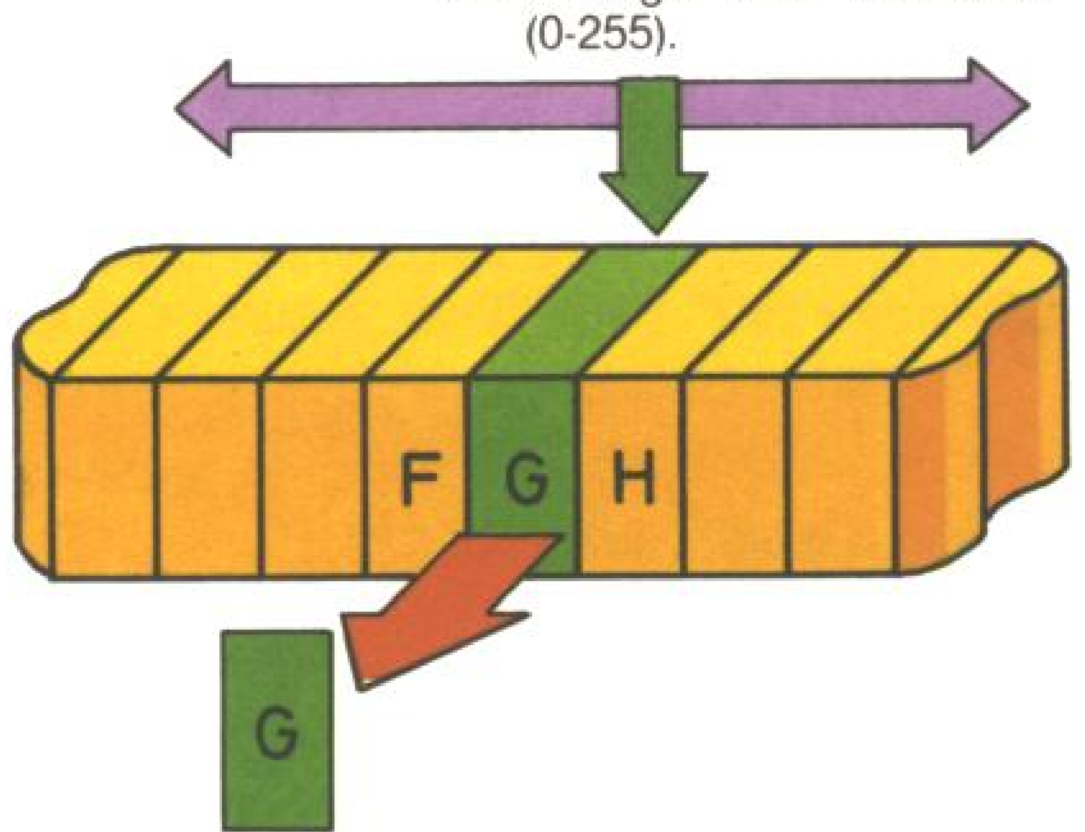
Resta A5H al acumulador situándose, si se trata de un TOKEN, en el rango 0-5BH. En este caso salta a PO-TOKENS (0C10H) para imprimirlo.

Si es un gráfico definido suma 15H para que su rango sea 0-15H, carga en BC (UDG) y salta a PO-CHAR-2 (Interior de PO-CHAR) para dibujarlo con PR-ALL.

PO-CHAR 0B65H 2917d

Busca en la tabla de caracteres el que corresponde pintar y entra en PR-ALL para hacerlo.

Datos de entrada: BC = Línea y columna inversos.
A = Código del caracter (0-255).



Nombre	**Hex.**	**Dec.**
PO-CHAR	0B65H	2917d
PR-ALL	0B7FH	2943d
PO-ATTR	0BDBH	3035d

Datos de salida : Ninguno.

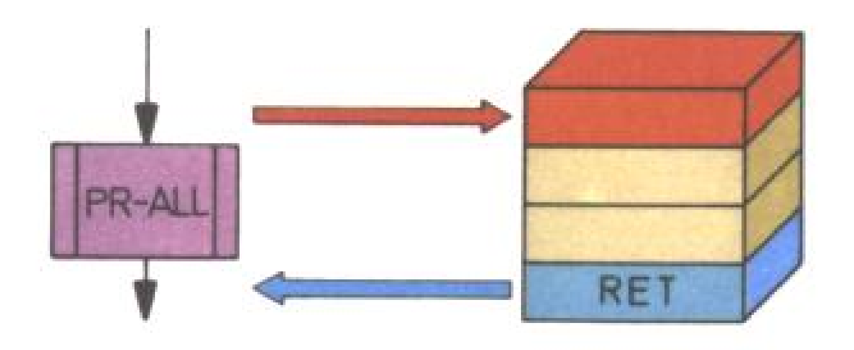
Registros modificados: Múltiples.
Variables modificadas: Las relativas al canal.

Rutinas que utiliza: **PR-ALL** (0B7FH).
Rutina usada por : **PO-ANY** (0B24H).

Observaciones: Esta rutina es muy útil pues permite escribir cualquier caracter de una tabla de 256. Para ello deberemos ejecutar la secuencia:

```
LD      A, Caracter
CALL    0B03H   ;PO-FETCH
CALL    0B65H   ;PO-CHAR
CALL    0ADCH   ;PO-STORE
```

Ello producirá un efecto similar a RST 10H.



PR-ALL 0B7FH 2943d

Rutina de impresión de un caracter con atributos.

En caso de no haber sitio en la pantalla produce un scroll.

Datos de entrada: BC = Línea y columna inversos.
HL = Dirección de esa posición.
A = Código del caracter (0-255).
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: Las relativas al canal.

Rutinas que utiliza: **COPY-BUFF** (0ECDH).
PO-SCR (0C55H).
PO-ATTR (0BDBH).
Rutina usada por : **PO-ANY** (0B24H).
PO-CHAR (0B65H).

PO-ATTR 0BDBH 3035d

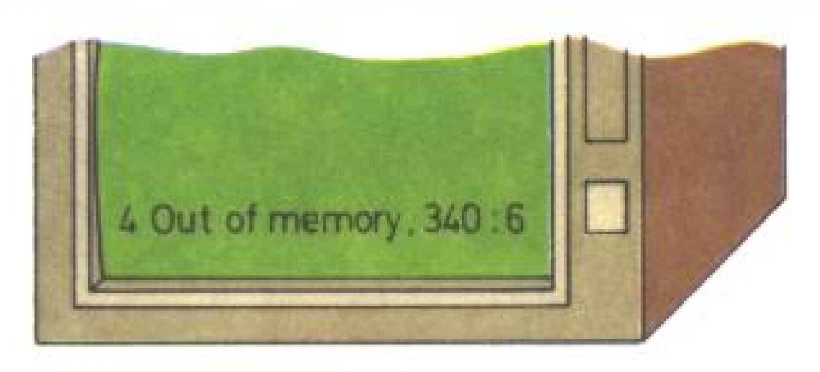
Pone los atributos a un caracter, según el que ya poseía y los valores determinados por ATTR-T, MASK-T y P-FLAG.

Datos de entrada: HL = Direc. en el archivo de imagen (alta resoluc.).
Datos de salida : HL = Dirección en el archivo de atributos (baja resolución).
D = ATTR-T E = MASK-T.

Registros modificados: AF,DE,HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **PR-ALL** (0B7FH).
PLOT (22DCH).





PO-MSG 0C0AH 3082d

Rutina de impresión de mensajes. Guarda un 0 en el byte alto del STACK como señal de «no poner espacio detrás» y salta a PO-TABLE.

Datos de entrada: A = Número de mensaje.
DE = Dirección de la tabla.

Observaciones: Cada mensaje debe ir precedido por un caracter con el bit 7 puesto a uno y su último caracter también.

PO-TOKENS 0C10H 3088d

Carga en DE 0095H (dirección de la tabla de TOKENS), guarda el número de mensje en el byte alto del STACK y entra en PO-TABLE.

Nombre	Hex.	Dec.	
PO-MSG	0C0AH	3082d	MENSAJES
PO-TOKENS	0C10H	3088d	
PO-TABLE	0C14H	3092d	
PO-SAVE	0C3BH	3131d	
PO-SEARCH	0C41H	3137d	
PO-SCR	0C55H	3157d	
TEMPS	0D4DH	3405d	

Datos de entrada: A = Número de TOKEN. (Cod.—A5H).

PO-TABLE 0C14H 3092d

Presenta un mensaje o TOKEN en pantalla con espacios delante o/y detrás si es necesario.

Rutinas que utiliza: **PO-SAVE** (0C3BH).
PO-SEARCH (0C41H).

Rutina usada por : **PO-MSG** (0C0AH).
PO-TOKENS (0C10H).

PO-SAVE 0C3BH 3131d

Rutina de salida de caracteres, salvando los registros BC,DE y HL.

Puede utilizarse en lugar de RST 10H para rutinas cíclicas.

PO-SEARCH 0C41H 3137d

Búsqueda de mensajes en una tabla.

Datos de entrada: A = Número de mensaje.
DE = Dirección de la tabla.
Datos de salida : DE = Dirección del mensaje.
Carry Flag (C) si no debe ser precedido de espacio. (A < 20H o el 1.^{er} caracter no es una letra).

Registros modificados: AF, DE.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : PO-TABLE.

Observaciones: Tanto el caracter precedente como el último de cada mensaje deben tener el bit 7 a 1.

PO-SCR 0C55H 3157d

Ver microficha M-17.

TEMPS 0D4DH 3405d

Esta importante rutina debe ejecutarse con las instrucciones de escritura en pantalla. Su misión consiste en copiar los atributos permanentes en los temporales.

Datos de entrada: Ninguno.
Datos de salida : HL = PFLAG A = (PFLAG).

Registros modificados: AF, HL.
Variables modificadas: ATTR-T,MASK-T, P-FLAG.

Rutinas que utiliza: Ninguna.
Rutina usada por : Los comandos de pantalla.

CLS 0D6BH 3435d

Rutina de borrado: Pone 0 en todos los bytes del «Display file», asigna a la parte superior de la pantalla el color de atributos permanentes (ATTR-P) y a la parte inferior el color del borde (BORDCR).

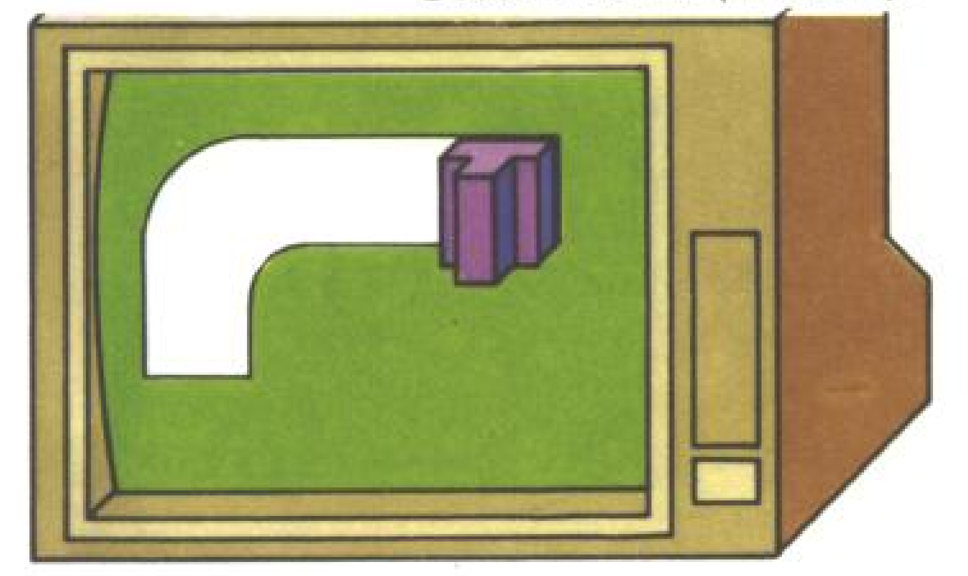
Datos de entrada: ATTRP.
Datos de salida : Punteros de pantalla e impresora en su comienzo.
HL = Dirección de comienzo de pantalla.
BC = Coordenadas de esa dirección.

Registros modificados: Múltiples.
Variables modificadas: Punteros de pantalla e impresora.

Rutinas que utiliza: **CL-ALL** (0DAFH).
TEMPS (0D4DH).
CL-LINE (0E44H).
CHAN-OPEN (1601H).
CL-SET (0DD9H).

Nombre	Hex.	Dec.	
CLS	0D6BH	3435d	COMANDO
CL-ALL	0DAFH	3503d	
CL-SET	0DD9H	3545d	
CL-SC-ALL	0DFEH	3582d	
CL-SCROLL	0E00H	3584d	
CL-LINE	0E44H	3652d	

Rutina usada por : Los comandos CLS y CLEAR
START-NEW (11CBH).



CL-ALL 0DAFH 3503d

Es la subrutina de CLS que borra la pantalla e inicializa los punteros.

CL-SET 0DD9H 3545d

Da la dirección del caracter cuyas coordenadas se encuentran en el par de registros BC o el número de columna en C si se trata de la impresora.

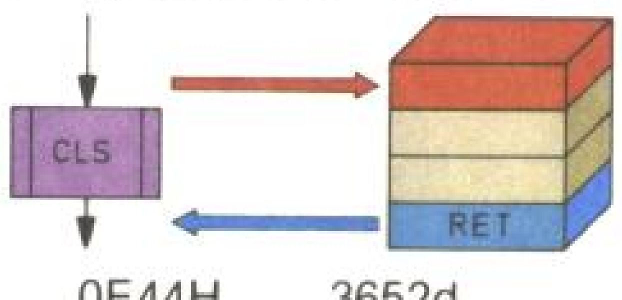
Datos de entrada: BC = Línea y columnas invertidas.
HL = Dirección del caracter.

Registros modificados: AF, BC, DE, HL.
Variables modificadas: Las relativas a la posición del cursor.

Rutinas que utiliza: PO-STORE (0ADCH).
Rutina usada por : Varios comandos.

Observaciones Dado que la rutina termina saltando a PO-STORE puede utilizarse para actualizar los punteros del cursor.

CL-SC-ALL **CL-SCROLL** Ver microficha M-17.



CL-LINE 0E44H 3652d

Borra de la pantalla el número de líneas indicado por el registro B contando desde la línea inferior.

Datos de entrada: B = Número de líneas.
B = Como entró.
C = 21H (33d): Columna 0.

Registros modificados: AF, BC, DE, HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: CL-ADDR.
CL-ATTR.
Rutina usada por : CL-ALL (0DAFH).
CL-SCROLL (0E00H).
AUTO-LIST (1795H).

PO-SCR 0C55H 3157d

Rutina de test de scroll: Se encarga de comprobar si es necesario hacerlo. Decrementa el contador de scrolls (SCR-CT) y, si éste llegó a 0, lo inicializa y escribe el mensaje «scroll?» esperando que sea pulsada una tecla.

Datos de entrada: BC = N.º de línea invertido.
Datos de salida : **BC** = Nueva línea y col.
HL = Direc. de esa posición.

Registros modificados: Múltiples.
Variables modificadas: SCR-CT y las relativas al cursor.

TEMPS Ver microficha M-15.

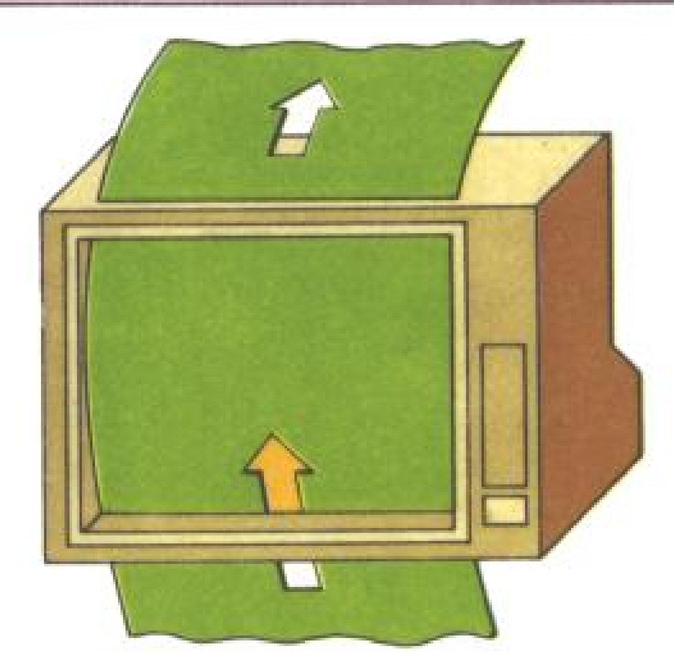
CLS **CL-ALL** **CL-SET** Ver M-16.

CL-SC-ALL 0DFEH 3582d

Rutina de Scroll. Es la entrada desde la pregunta «Scroll?». Hace un desplazamiento hacia arriba de toda la pantalla.

Carga en B 17H (23d) y entra en CL-SCROLL.

Nombre	Hex.	Dec.
PO-SCR	0C55H	3157d
TEMPS	0D4DH	3405d
CLS	0D6BH	3435D
CL-ALL	0DAFH	3503d
CL-SET	0DD9H	3545d
CL-SC-ALL	0DFEH	3582d
CL-SCROLL	0E00H	3584d
CL-ATTR	0E88H	3720d



CL-SCROLL 0E00H 3584d

Rutina de Scroll parcial (continuación de CL-SC-ALL). Produce un desplazamiento hacia arriba del número de líneas indicado por el registro B empezando a contar desde abajo.

Termina entrando en CL-LINE (0E44H) para borrar la línea inferior que quedó repetida.

Es llamada al hacer un cambio de línea si se está trabajando en la parte inferior de la pantalla.

Datos de entrada: Ninguno.
Datos de salida : Ninguno.

Registros modificados: AF, BC, DE, HL.
Variables modificadas: Relativas a la pantalla.

Rutinas que utiliza: **CL-ADDR** (0E9BH).
CL-ATTR (0E88H).
CL-LINE (0E44H).
Rutina usada por : **PO-SCR** (0C55H).

CL-ATTR 0E88H 3720d

Esta rutina tiene dos funciones:

a) Proporciona la dirección de un caracter en el archivo de atributos a partir del «noveno byte» en el archivo de imagen.

b) Informa del número de caracteres que hay desde esa línea al final de la pantalla.

Datos de entrada: HL = Dirección del 9.° byte.
B = N.° de línea invertido.
C = 0.
Datos de salida : DE = Dirección del atributo.
BC = HL = 32*B.

Registros modificados: AF, BC, DE, HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **CL-LINE** (0E44H).
CL-SCROLL (0E00H).

Observaciones: Se entiende por «9.° byte» el primero más H incrementado en 8.

CL-ADDR 0E9BH 3739d

Obtiene la dirección en el archivo de imagen del primer caracter de la línea especificada por el registro B.

Datos de entrada: B = N.° de línea invertido.
Datos de salida : HL = Dirección del 1.er caracter.
D = Número de línea.
A = H.

Registros modificados: A, B, H, L.
Variables modificadas: Ninguna.

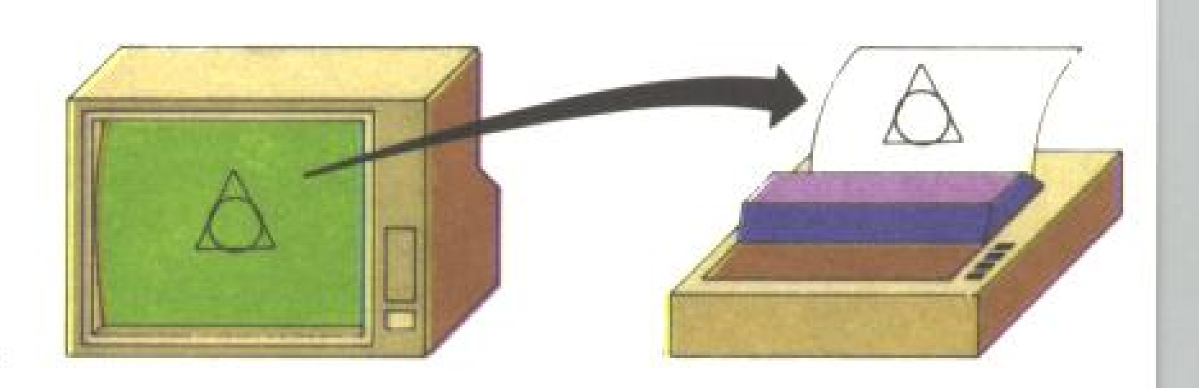
Rutinas que utiliza: Ninguna.
Rutina usada por : **PO-SCR** (0C55H).
CL-SET (0DD9H).
CL-SCROLL (0E05H).
CL-LINE (0E44H).

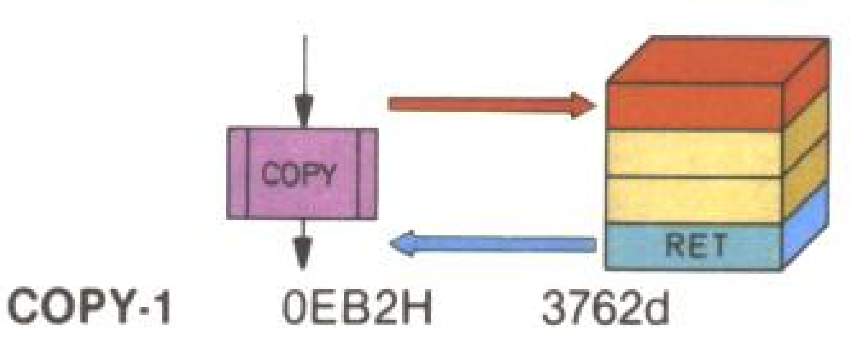
COPY 0EACH 3756d

Rutina del comando COPY: deshabilita las interrupciones, da a B el valor 175 (líneas de la parte superior de la pantalla) y a HL la dirección del comienzo de la pantalla (4000H).

Posteriormente entra en COPY-1.

Nombre	Hex.	Dec.	
CL-ADDR	0E9BH	3739d	
COPY	0EACH	3756d	COMANDO
COPY-1	0EB2H	3762d	
COPY-BUFF	0ECDH	3789d	
CLEAR-PRB	0EDFH	3807d	
COPY-LINE	0EF4H	3828d	





COPY-1 0EB2H 3762d

Bucle de escritura en impresora del comando COPY. Para que funcione correctamente han de estar deshabilitadas las interrupciones y encontrarse en el registro B el número de líneas en alta resolución que se desea copiar.

Datos de entrada: B = Número de líneas por copiar.
HL = Dirección del primer byte.
Datos de salida : HL = Ultimo byte copiado + 1.

Registros modificados: AF, BC, DE, HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: **COPY-LINE** (0EF4H).
Rutina usada por : El comando **COPY.**

Observaciones: Para copiar la totalidad de la pantalla debe hacerse:

```
DI
LD    B,192
LD    HL,16384
CALL  3762
```

COPY-BUFF 0ECDH 3789d

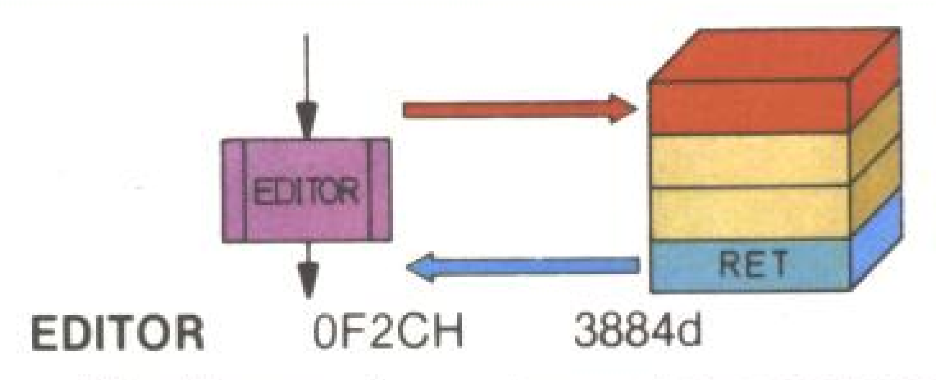
Rutina utilizada por el comando LPRINT: Vuelca a la impresora el contenido del Buffer. Utiliza 8 veces la rutina COPY-LINE.

CLEAR-PRB 0EDFH 3807d

Limpia el buffer de la impresora y actualiza los punteros mediante las rutins CL-SET (0DD9H) y PO-STORE (0ADCH).

COPY-LINE 0EF4H 3828d

Copia en impresora una línea de pixels en alta resolución. Para ello utiliza el port 251 (FBH).



EDITOR 0F2CH 3884d

El editor es llamado en dos ocasiones:

a) En la rutina principal de ejecución MAIN-2 (12ACH) para introducir un comando o una línea Basic.

b) En la rutina del comando INPUT (2089H) para introducir un dato en una variable.

El Editor atiende a los comandos de edición, recibe información por el canal K (normalmente del teclado, mediante la rutina KEY-INPUT) y la guarda en el espacio de trabajo (WORK-SP) si se trata de una sentencia INPUT, o en el área de edición si se está introduciendo una línea Basic o un comando directo.

Sólo se sale del Editor mediante la tecla ENTER, pues incluso posee su propia rutina en caso de error (ED-ERROR).

Nombre	Hex.	Dec.
EDITOR	0F2CH	3884d
ADD-CHAR	0F81H	3969d
ED-KEYS	0F92H	3986d
ED-EDIT	0FA9H	4009d
ED-DOWN	0FF3H	4083d
ED-LEFT	1007H	4103d
ED-RIGHT	100CH	4108d
ED-DELETE	1015H	4117d
ED-IGNORE	101EH	4126d
ED-ENTER	1024H	4132d
ED-EDGE	1031H	4145d
ED-UP	1059H	4185d
ED-SYMBOL	1076H	4214d
ED-GRAPH	107CH	4220d
ED-ERROR	107FH	4223d
CLEAR-SP	1097H	4247d

ADD-CHAR 0F81H 3969d

Agrega un nuevo carácter en el espacio de trabajo o el área de edición.

ED-KEYS 0F92H 3986d

Rutina que gestiona la tabla de saltos a las rutinas de control: ED-EDIT (Caps + 1), ED-DOWN (cursor bajo), ED-LEFT (cursor izquierda), ED-RIGHT (cursor derecha), ED-DELETE (Caps + 0 ; borra carácter), ED-ENTER, ED-UP (cursor arriba), ED-SYMBOL (Caps + Symbol) shift), y ED-GRAPH (Casps + 9).

ED-IGNORE 101EH 4126d

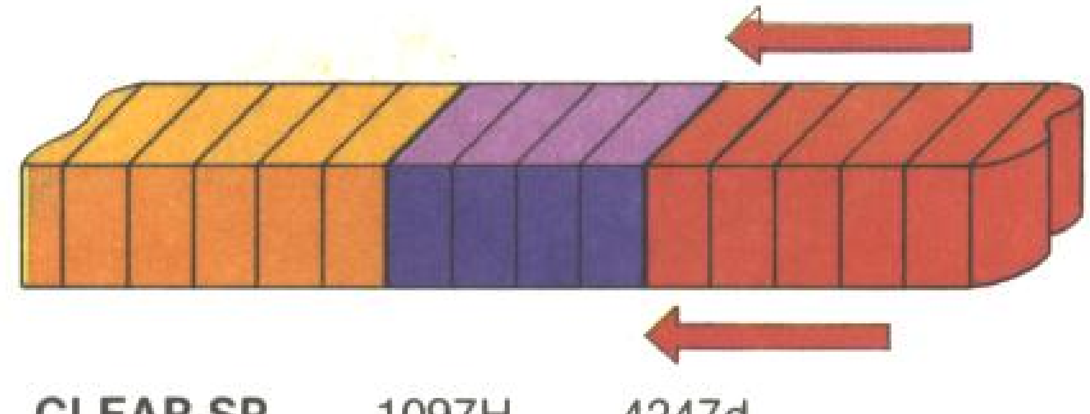
Ignora los dos caracteres siguientes a AT o TAB.

ED-EDGE 1031H 4145d

Controla que el cursor no sobrepase el comienzo de línea al borrar o retroceder, también le impide colocarse entre un código de control y sus parámetros.

ED-ERROR 10F7H 4223d

Anula el código de error y tras producir un sonido de aviso vuelve al editor.



CLEAR-SP 1097H 4247d

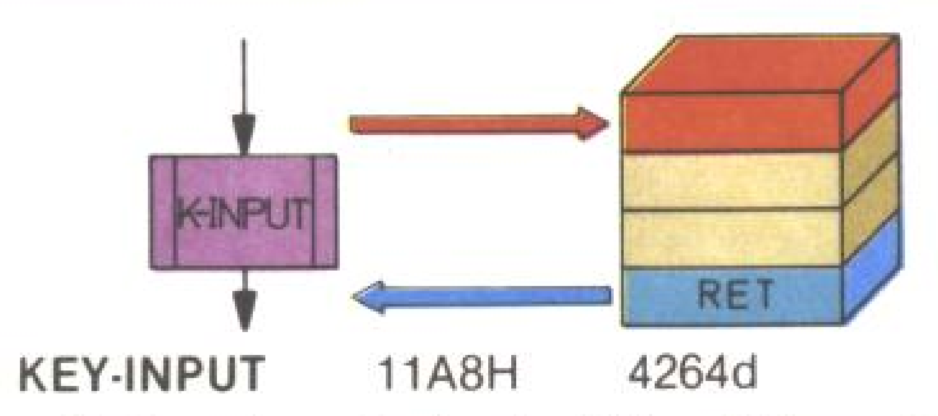
Borra el espacio de trabajo o el área de edición (según indique el bit 5 de la variable FLAGX).

Datos de entrada: Ninguno.
Datos de salida : Ninguno.

Registros modificados: AF,BC,DE.
Variables modificadas: K-CUR,MODE y los punteros del Basic.

Rutinas que utiliza: **SET-HL** (1190H). **RECLAIM1** (19E5H).
Rutina usada por : **ED-EDIT** (0FA9H). **MAIN-5** (133CH).





KEY-INPUT 11A8H 4264d

Rutina de entrada de datos del canal K. La rutina INPUT-AD (15E6H) lee en (CURCHL + 2) esta dirección cuando ha sido abierto el canal 1 (K) con la rutina **CHAN-OPEN** (1601H).

Devuelve en el acumulador el código de la última tecla pulsada. Si el bit 3 de TV-FLAG indica que el modo ha cambiado, llama a la rutina ED-COPY.

Datos de entrada: Ninguno.
Datos de salida : A = Tecla pulsada.

Registros modificados: Múltiples.
Variables modificadas: Múltiples.

Rutinas que utiliza: **ED-COPY** (111DH).
CLS-LOWER (0D6EH).

Nombre	Hex.	Dec.
KEY-INPUT	10A8H	4264d
ED-COPY	111DH	4381d
SET-HL	1190H	4496d
SET-DE	1195H	4501d
REMOVE-FP	11A7H	4519d

Rutina usada por : El canal K para entrada de datos.

Observaciones: Esta rutina no inspecciona el teclado, sino que lee la variable del sistema LAST-K. Para que sea leído el teclado han de estar habilitadas las interrupciones.

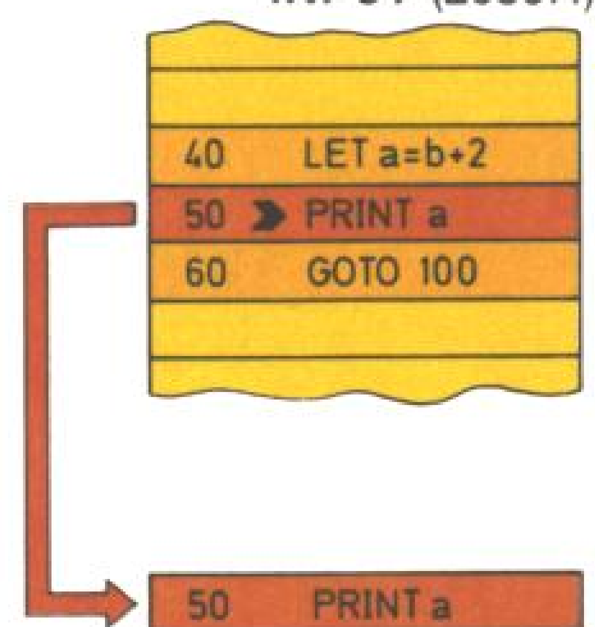
ED-COPY 111DH 4381d

Escribe en la parte inferior de la pantalla el contenido del área de trabajo o la zona de edición según indique el bit 5 de la variable FLAGX.

Datos de entrada: bit 5, (FLAGX).
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: Múltiples.

Rutinas que utilza : **TEMPS** (0D4DH).
SET-DE (1195H).
OUT-LINE (187DH).
OUT-CURS (18E1H).
PRINT-OUT (09F4H).
BEEPER (03B5H).
CL-SET (0DD9H).
Rutina usada por : **KEY-INPUT** (10A8H).
INPUT (2089H).



SET-HL 1190H 4496d

Sitúa en HL el principio, y en DE el final, del espacio de trabajo o el área de edición, según indique el bit 5 de la variable FLAGX.

Datos de entrada: Bit 5, (FLAGX).
Datos de salida : HL = Comienzo del buffer.
DE = Final del buffer.

Registros modificados: HL,DE.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **CLEAR-SP** (1097H).

SET-DE 1195H 4501d

Continuación de **SET-HL;** igual que la rutina anterior pero sólo para el final del área.

REMOVE-FP 11A7H 4519d

Coloca en la pila todos los números en coma flotante de una línea Basic que se está interpretando.

NEW 11B7H 4535d

Rutina del comando NEW. Comprueba e inicializa la memoria hasta la dirección señalada por RAMTOP (normalmente asignada por el comando CLEAR).

Mantiene los valores de las variables PRAMPT, RASP, PIP, UDG, y RAMTOP e inicializa el resto de las variables.

Datos de entrada: RAMTOP.
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: Todas menos las arriba indicadas.

Rutinas que utiliza: **CLEAR-PRB** (0EDFH).
CLS (0D6BH).
PO-MSG (0C0AH).
MAIN-1 (12A9H).
Rutina usada por : El comando **NEW.**

Observaciones: Esta rutina también inicializa el Stack, por lo que es imposible volver de ella. Termina entrando en el bucle principal (MAIN-1).

Nombre	Hex.	Dec.	
NEW	11B7H	4535d	COMANDO
START/NEW	11CBH	4555d	START
MAIN-EXEC	12A2H	4770d	
MAIN-1	12A9H	4777d	BUCLE
MAIN-2	12ACH	4780d	PRINCIPAL
MAIN-3	12CFH	4815d	
MAIN-4	1303H	4867d	(ERR-SP)
MAIN-5a9	133CH	4924d	
REP-MESS	1391H	5009d	MENSAJES
REPORT-G	1555H	5461d	ERROR
MAIN-ADD	155DH	5469d	

START/NEW 11CBH 4555d

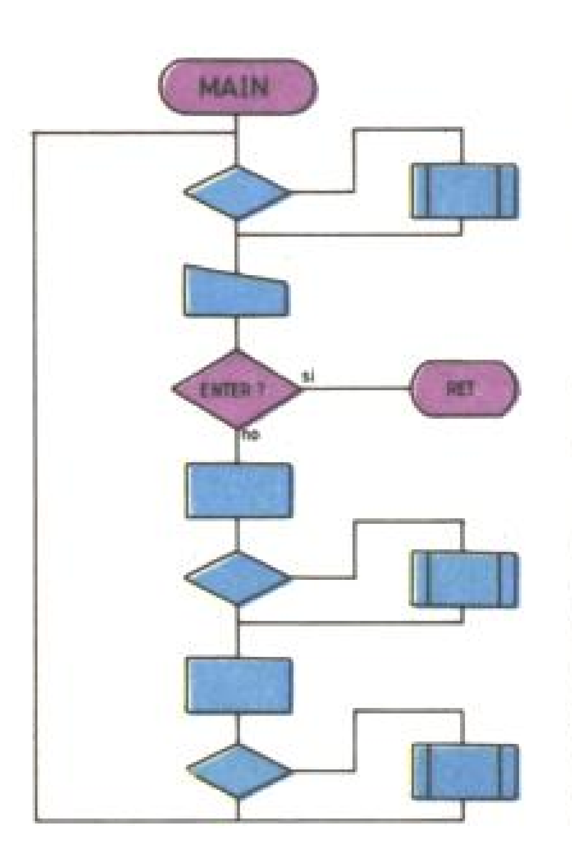
Rutina de inicialización; se ejecuta al hacer un RESET o al conectar el ordenador llamada por RST 0.

Comprueba e inicializa toda la memoria.

BUCLE PRINCIPAL (MAIN)

Las direcciones de memoria 12A2H a 15AE constituyen un bucle en torno al cual discurre todo el funcionamiento del ordenador.

Para ello utiliza convenientemente las siguientes rutinas:



AUTO-LIST	1795H
SET-MIN	16B0H
CHAN-OPEN	1601H
EDITOR	0F2CH
LINE-SCAN	1B17H
REMOVE-FP	11A7H
LINE-NO	19FBH
CL-ALL	0DAFH
CLS-LOWER	0D6EH
LINE-RUN	1B8AH
COPY-BUFF	0ECDH
OUT-CODE	15EFH
PRINT-A-1	0010H
PO-MSG	0C0AH
OUT-NUM-1	1A1BH
CLEAR-SP	1097H

● Las diferente partes de la rutina son:

MAIN-EXEC: Produce un listado automático.
MAIN-1: Borra las zonas de trabajo.
MAIN-2: Abre el canal K y llama al editor.
MAIN-3: Ejecuta una línea o comando directo.
MAIN-4: Dirección de retorno de la ejecución de un programa o comando. También es la señalada por (ERR-SP) para retorno de error.
MAIN-5 a MAIN-9: Escriben el mensaje correspondiente y ajustan las variables SUBPPC, OLDPPC y OSPPC.

REP-MESS 1391H 5009d

Tabla de los mensajes de error. El carácter precedente y el último de cada mensaje tienen el bit 7 a 1.

MAIN-ADD 155DH 5469d

Esta rutina añade o sustituye una nueva línea en el listado. Es llamada por el bucle principal desde MAIN-3, una vez comprobada la sintaxis.

INIT-CHAN 15AFH 5551d

Tabla de las direcciones iniciales para los canales «K», «S», «R» y «P» para comunicación respectivamente con el teclado y parte inferior de la pantalla, la pantalla principal, el espacio de trabajo y la impresora.

CANAL	SALIDA	ENTRADA
K	09F4H PRINT-OUT	10A8H KEY-INPUT
S	09F4H PRINT-OUT	15C4 ERROR-J
R	0F81H ADD-CHAR	15C4 ERROR-J
P	09F4H PRINT-OUT	15C4 ERROR-J

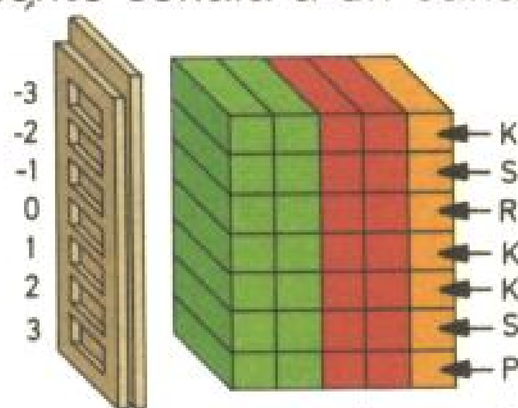
Estas direcciones son almacenadas en la zona señalada por CHANS mediante la rutina START/NEW (11CBH) situando como marca de final el código 0.

Nombre	Hex.	Dec.
INIT-CHAN	15AFH	5551d
INIT-STRM	15C6H	5574d
WAIT-KEY	15D4H	5588d
INPUT-AD	15E6H	5606d

INIT-STRM 15C6H 5574d

Tabla inicialización de las siete corrientes de información: —3 (FDH) a +3.

Cada corriente señala a un canal:



Estos punteros son cargados en las primeras direcciones de la variable STRMS por la rutina START/NEW (11CBH).

WAIT-KEY 15D4H 5588d

Bucle de espera hasta que llegue un carácter por el canal de entrada. (Normalmente el teclado).

Datos de entrada: BIT 5,(TV-FLAG).
Datos de salida : Según la rutina de INPUT; Generalmente A = Código del carácter.

Registros modificados: Según canal usado.
Variables modificadas: Las relativas al canal.

Rutinas que utiliza: **INPUT-AD** 15E6H.
Rutina usada por : **SA-CONTRL** 0970H.
PO-SCR 0C55H.
EDITOR 0F2CH.

Observaciones: El bucle termina cuando la rutina de entrada devuelva el flag de Carry. Si devuelve NC y NZ se produce el error 8. El bucle continúa mientras esté alzado el flag Z.

El bit 5 de FLAGS a 1 indica que la parte inferior de la pantalla ha de ser borrada.

INPUT-AD 15E6H 5606d

Llama a la rutina de INPUT correspondiente al canal en curso: la señalada por (CURCHL) + 2.

Es preservado el registro HL'.

Datos de entrada: CURCHL.
Datos de salida : Según el canal.

Registros modificados: Según canal usado.
Variables modificadas: Las relativas al canal.

Rutinas que utiliza: **CALL-SUB** 15F7H.
CALL-JUMP 162CH.
Rutina usada por : **WAIT-KEY** 15D4H.
read-in(CALCULADOR) 3645H.

Observaciones: Normalmente es usado el canal K que envía a la rutina KEY-INPUT. En tal caso los datos de salida son:

Carry: Código aceptable.
Z y NC: No tecla pulsada.
NC y NZ: Pulsación incorrecta.

OUT-CODE 15EFH 5615d

Envía por el canal en curso una cifra: Incrementa en 48 el valor del acumulador y entra en PRINT A-2.

PRINT-A-2 15F2H 5618d

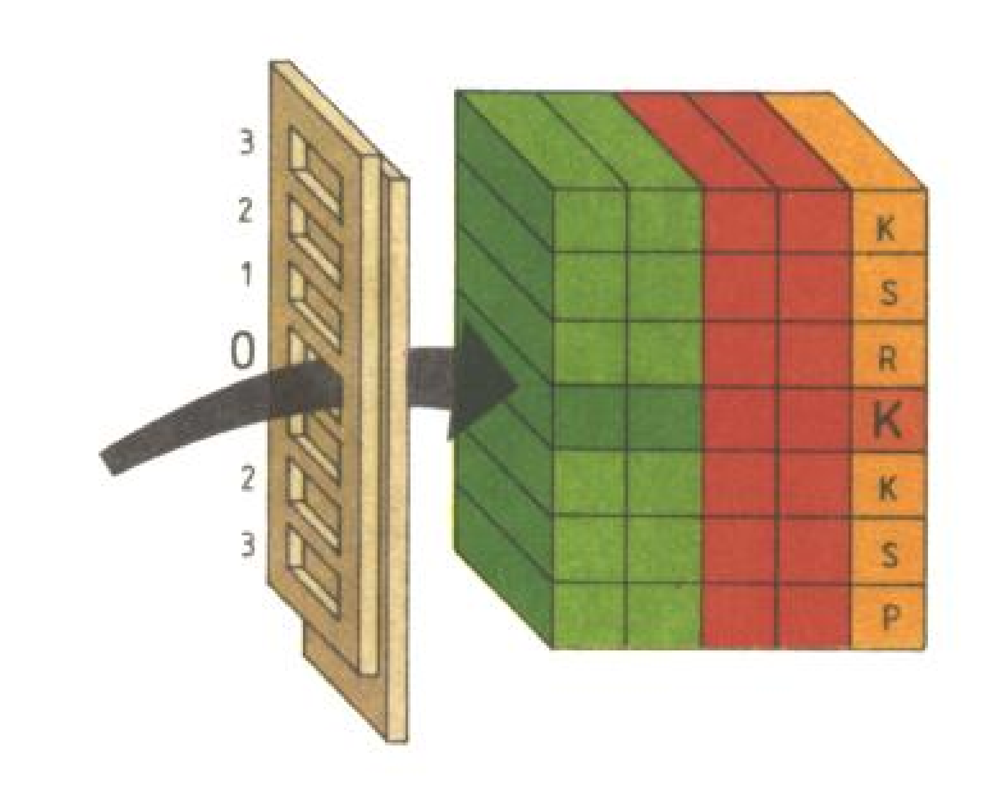
Envía el carácter contenido en A por el canal en curso, señalado por CURCHL al ser abierto por CHAN-OPEN (1601H). Es la rutina utilizada por RST 10H (ver microficha M-2).

CHAN-OPEN 1601H 5633d

Esta rutina se encarga de abrir uno de los canales de información. Si el canal abierto es K, S o P se efectúa un salto a la correspondiente rutina que ajusta TV-FLAG, FLAGS y FLAGS2 y ATTRT.

Datos de entrada: A = Número del canal.
Datos de salida : CURCHL apuntando al canal abierto.
Error O si la corriente no existe (marcada con 0).

Nombre	Hex.	Dec.	
OUT-CODE	15EFH	5615d	
PRINT-A-2	15F2H	5618d	SALIDA
CHAN-OPEN	1601H	5633d	ABRE CANAL
CHAN-FLAG	1615H	5653d	
CALL-JUMP	162CH	5676d	CALL INDIR.



Registros modificados: A,C,HL,DE.
Variables modificadas: CURCHL, TV-FLAG, FLAGS, FLAGS2, ATTR-T.

Rutinas que utiliza: **INDEXER** 16DCH.
CHAN-FLAG 1615H.
Rutina usada por : Múltiples comandos.

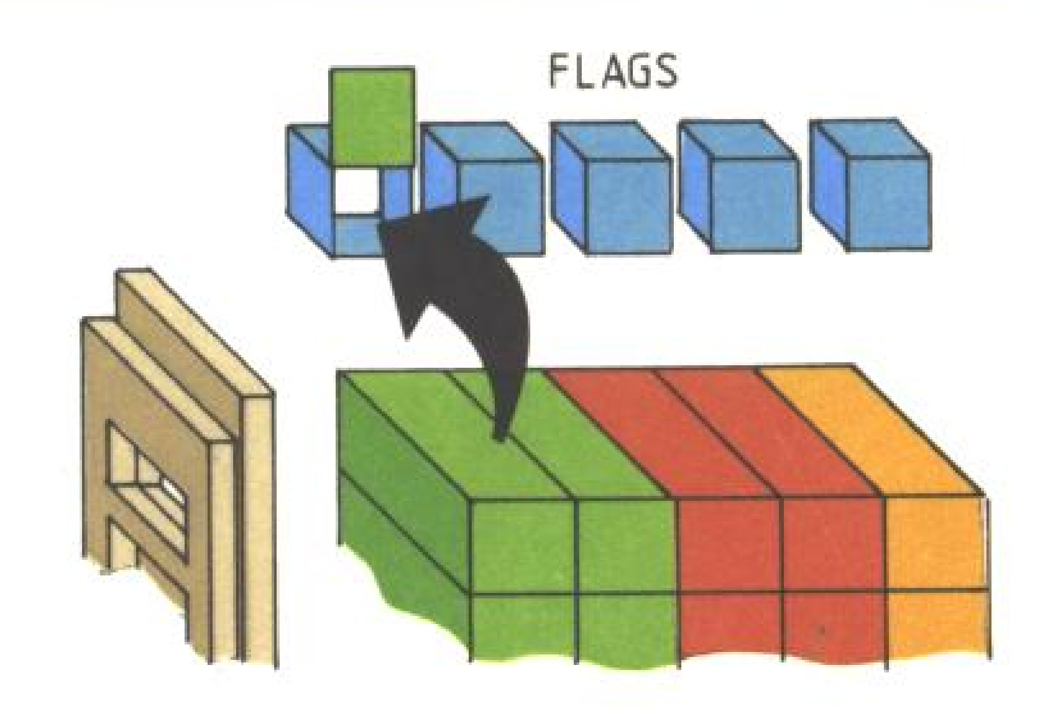
Observaciones: Por ejemplo, si se desea que RST 10H envíe los caracteres a la parte superior de la pantalla deberá hacerse previamente:

```
LD     A,2
CALL   5633
```

CHAN-FLAG, CHAN-K, CHAN-S, CHAN-P

Los tres canales K, S y P utilizan la misma rutina de salida de datos: PRINT-OUT (09F4H).

Para distinguir de qué canal se trata estas rutinas utilizan el BIT 0 de TV-FLAG, el BIT 1 de FLAGS y el 4 de FLAGS2. Al abrir los canales K y S es llamada la rutina TEMPS (0D4DH).



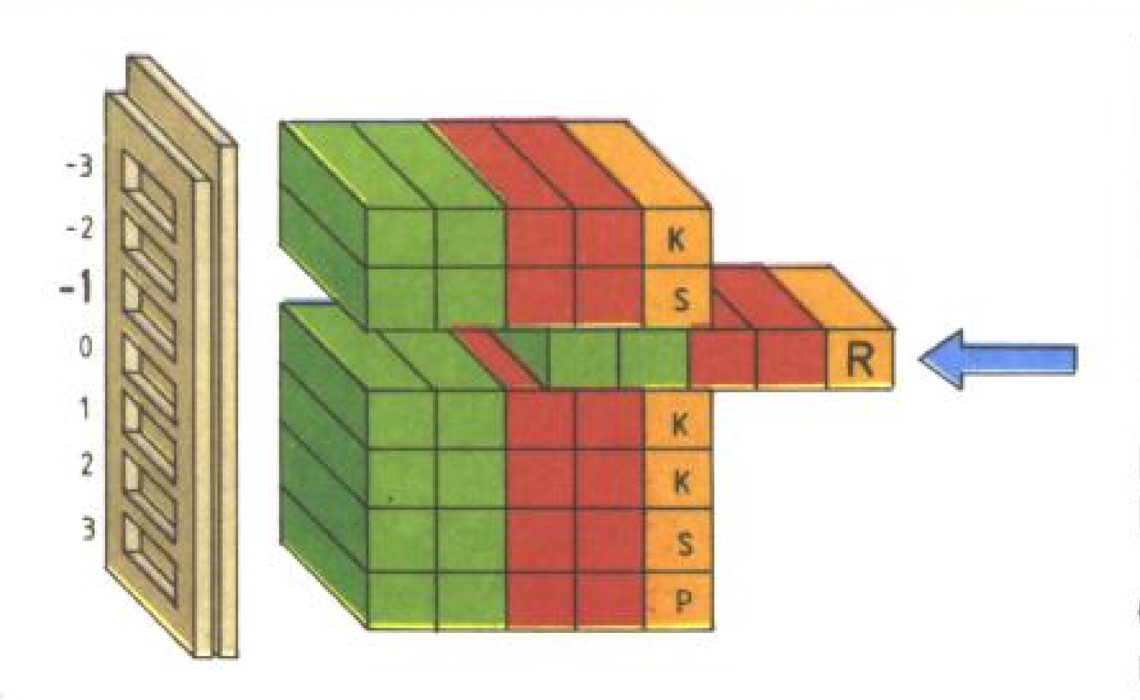
CALL-JUMP 162CH 5676d

Esta importante rutina que sólo consta de una instrucción «JP (HL)» sirve para implementar la instrucción inexistente «CALL (HL)». Es imprescindible para utilizar tablas de llamadas a diferentes subrutinas.

```
Ejemplo:  LD     HL,RUT   Equivale a:
          CALL   5676          CALL RUT
```

Close, Memoria I





Nombre	Hex.	Dec.	
CLOSE	16E5H	5861d	COMANDO
ONE-SPACE	1652H	5714d	
MAKE-ROOM	1655H	5717d	ABRE MEM.
POINTERS	1664H	5732d	

CLOSE 16E5H 5861d

Rutina para cerrar una corriente (stream).

Datos de entrada: Número de la corriente en el STACK del calculador.
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: STREAMS y las del calculador.

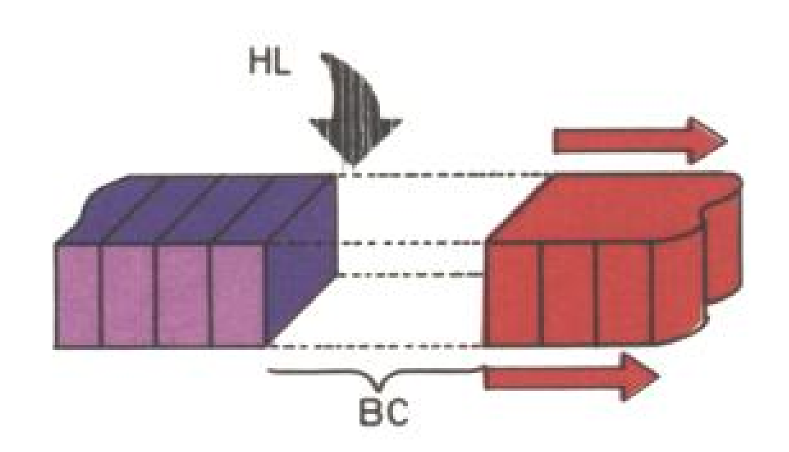
Rutinas que utiliza: STK-TO-A.
Rutina usada por : El comando CLOSE#.

Observaciones: Las corrientes 0 a 3 no se cierran sino que le son asignados los canales iniciales K, K, S y P.

ONE-SPACE 1652H 5714d

Abre un hueco de un byte en cualquier parte de las zonas dinámicas bajas (ver G-27) y ajusta los punteros con la nueva posición de la memoria. Es usada por ADD-CHAR (0F81H).

Carga en BC 1 y entra en MAKE-ROOM (1655H).



MAKE-ROOM 1655H 5717d

Abre un hueco de un número de bytes especificado por el par BC en cualquier parte de las zonas dinámicas bajas (Ver G-27) y ajusta los punteros con la nueva posición de la memoria.

Datos de entrada: BC: Número de bytes.
HL: Dirección.
Datos de salida : HL: Descrementado en 1.
DE: Ultimo byte nuevo.

Registros modificados: BC, DE, HL.
Variables modificadas: Los punteros del BASIC.

Rutinas que utiliza: **TEST-ROOM** 1F05H.
POINTERS 1664H.
Rutina usada por : Múltiples comandos.



POINTERS 1664H 5732d

Incrementa en el valor de BC el contenido de todos los punteros del BASIC (ver microficha G-30) que señalen más allá que el par HL.

Datos de entrada: BC = longitud.
HL = Dirección.
Datos de salida : DE = Antiguo STKEND.
HL = Como entró.
BC = Antiguo STKEND—HL.

Registros modificados: BC,DE.
Variables modificadas: Punteros del BASIC.

Rutinas que utiliza: Ninguna.
Rutina usada por : **MAKE-ROOM** 1655H.
RECLAIM 19E5H.

LINE-NO 1695H 5781d

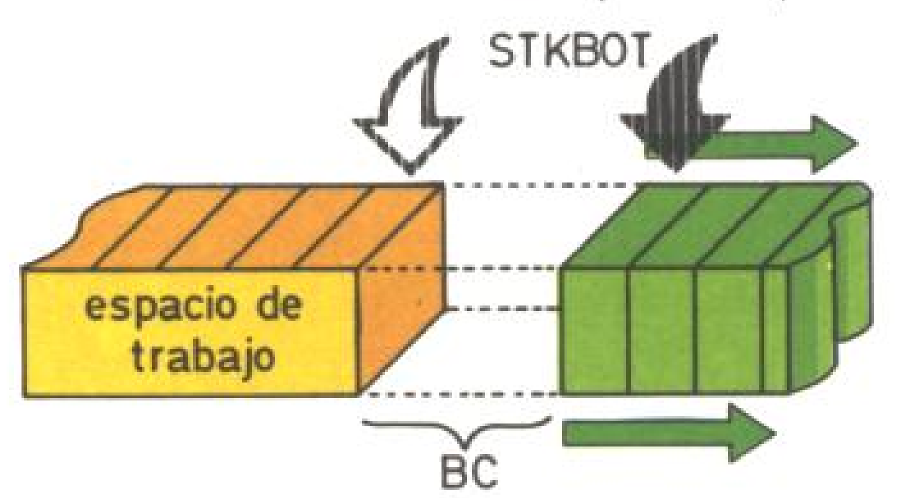
Sitúa en el par de registros DE el número de línea señalado por HL si es menor de 16384, o el señalado por DE si es menor que esta cantidad, o, en caso contrario 0.

Registros modificados: A,HL,DE.

RESERVE 169EH 5790d

Abre BC espacios en la zona de trabajo. Utiliza MAKE-ROOM (1655H).

No funciona aisladamente, sino como continuación de la rutina RST 30 (Ver M-3).



LINE-NO	1695H	5781d	
RESERVE	169EH	5790d	
SET-MIN	16B0H	5808d	CIERRA M.
SET-WORK	16BFH	5823d	CIERRA M.
SET-STK	16C5H	5829d	CIERRA M.
INDEXER	16DCH	5852d	

SET-MIN 16B0H 5808d

Esta rutina anula la zona de edición, el espacio de trabajo y el stack del calculador.

MEM toma el valor 5C92H (MEMBOT).

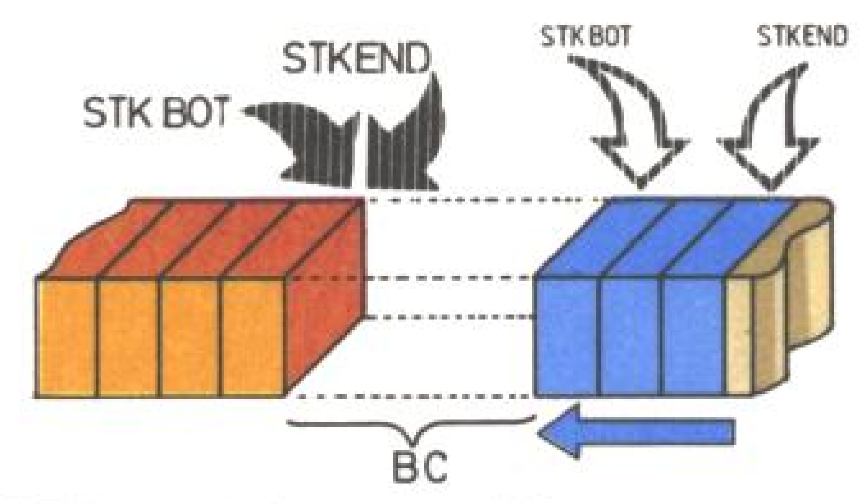
Datos de entrada: Ninguno.
Datos de salida : HL = nuevo (STKEND).

Registros modificados: HL.
Variables modificadas: K-CUR, WORKSP, STKBOT, STKEND Y MEM.

Rutinas que utiliza: Ninguna.
Rutina usada por : **MAIN-1** (12A9H).
MAIN-4 (1303H).

SET-WORK 16BFH 5823d

Continuación de SET-MIN; anula el espacio de trabajo y el stack del calculador respetando la zona de edición.



SET-STK 16C5H 5829d

Ultima parte de SET-MIN; elimina sólo el stack del calculador. Es utilizada por ERROR-3 (0055H), continuación de RST8.

INDEXER 16DCH 5852d

Localiza un byte en una tabla que comienza en la dirección señalada por HL hasta la marca de final «0».

Datos de entrada: HL = Dirección de comienzo de búsqueda.
C = Dato a buscar.

Datos de salida : HL señalando 1 byte más adelante del buscado o del final.
Carry si se encontró el dato.

Registros modificados: HL,A.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **CHAN-FLAG** 1615H.
CLOSE 16E5H.
OPEN 1736H.
SCANNING 24FBH.

OPEN 1736H 5942d

Abre una corriente hacia uno de los canales K, S o P.

Datos de entrada: Número de la corriente y nombre del canal en el STACK del calculador.
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: STREAMS y STKEND.

Rutinas que utiliza: **FP-CALC**0028H.
STK-FETCH 2BF1H.
INDEXER 16DCH.
Rutina usada por : El comando OPEN#.

CAT-ETC 1793H 6035d

Los comandos CAT, ERASE, FORMAT y MOVE no están implementados en la ROM ordinaria. Se produce un mensaje de error «O».

Nombre	Hex.	Dec.	
OPEN	1736H	9542d	
CAT-ETC	1793H	6035d	
AUTO-LIST	1795H	6037d	
LLIST	17F5H	6133d	COMANDO
LIST	17F9H	6137d	COMANDO
LIST-ALL	1835H	6197d	
OUT-LINE	1855H	6229d	

AUTO-LIST 1795H 6037d

Muestra la página del listado donde se encuentra el cursor de línea. Es usada por el EDITOR (0F2CH) y el bucle principal al añadir una nueva línea MAIN-EXEC (12A2H).

LLIST 17F5H 6133d

Listado por impresora; Abre el canal 3 y entra en la rutina LIST.

LIST 17F9H 6137d

Listado por cualquier canal. El número del canal es leído mediante sucesivas llamadas a GET-CHAR (0018H) y debe estar escrito en ASCII y señalado por CH-ADD.

Una forma más cómoda de hacer un listado desde código máquina es abrir el canal que se desee con CHAN-OPEN (1601H) y llamar a la rutina a la dirección **182DH** teniendo en HL el número de línea desde donde se desea listar.

Ejemplo:
```
LD    A,2
CALL  CHAN-OPEN
LD    HL,Línea
CALL  182DH
```

LIST-ALL 1835H 6197d

Rutina de listado común para AUTO-LIST, LLIST y LIST.

OUT-LINE 1855H 6229d

Rutina de impresión de una línea del listado. HL debe contener la dirección de ésta.

Datos de entrada: HL = Dirección de la línea.
Datos de salida : HL = Comienzo de la línea siguiente.
D = 3EH si es la línea actual.
DE = 0 si la actual es anterior.
DE = 1 si la actual es posterior.

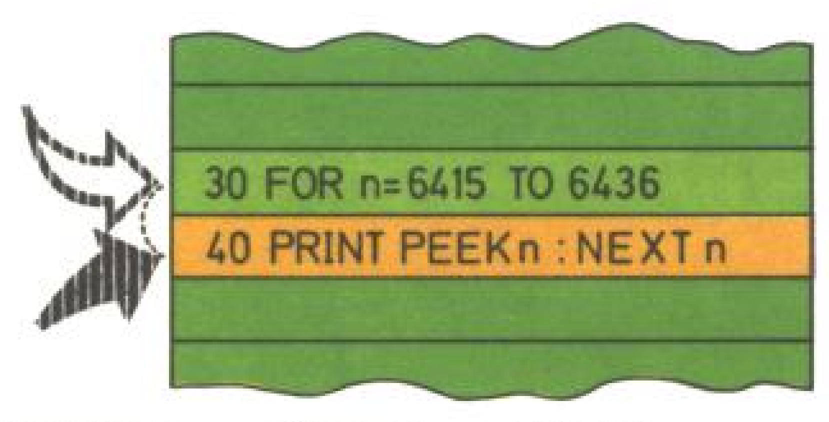
Registros modificados: Múltiples.
Variables modificadas: Múltiples.

Rutinas que utiliza: **CP-LINES** 1980H.
Rutina usada por : **PRINT-A-1** 0010H.
ED-EDIT OFA9H
LIST-ALL 1835H

Observaciones: Para listar una línea puede hacerse:

```
LD    HL,número de línea.
CALL  6510 ; LINE-ADDR.
CALL  6229 ; OUT-LINE.
```





Nombre	Hex.	Dec.
LN-FETCH	190FH	6415d
LINE-ADDR	196EH	6510d
CP-LINES	1980H	6528d
EACH-STMT	198BH	6539d
NEXT-ONE	19B8H	6584d

LN-FECTH 190FH 6415d

Incrementa el puntero al listado BASIC.

Datos de entrada: HL = S-TOP o E-PPC.
BIT 5 (FLAGX).
Datos de salida : DE = Direc. línea siguiente.
Si BIT 5,(FLAGX) = 0 es actualizada la variable.

Registros modificados: A,BC,DE,HL.
Variables modificadas: La señala por HL.

Rutinas que utiliza: **LINE-ADDR** 196EH.
LINE-NO 1695H.
Rutina usada por : **LIST-ALL** 1835H.
EDITOR 0F2CH.

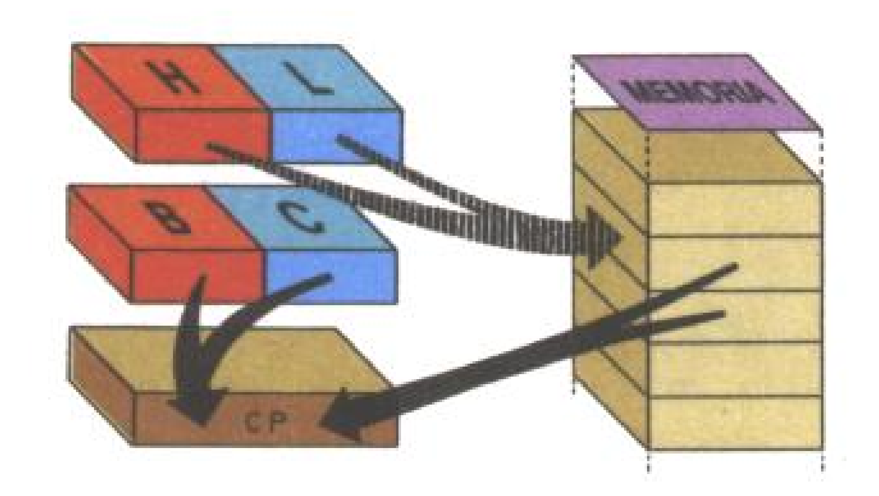
LINE-ADDR 196EH 6510d

Calcula la dirección de una línea BASIC o la primera línea posterior si ésta no existe.

Datos de entrada: HL: Número de línea.
Datos de salida : HL: Dirección de ésta o la más próxima.
DE: Dirección línea anterior.
Z flag: Si existe la línea.

Registros modificados: A,BC,DE,HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: **CP-LINES** 1980H.
NEXT-ONE 19B8H.
Rutina usada por : Múltiples comandos.



CP-LINES 1980H 6528d

Compara BC y (HL),(HL + 1) devolviendo los flags correspondientes. Sólo modifica A.

EACH-STMT 198BH 6539d

Localiza el comienzo de la instrucción dentro de una línea BASIC indicada por el registro D o la que comience por el TOKEN indicado por el registro E a partir de (CH-ADD). Ver LOOK-PROG (1D86H) en microficha M-37.

La dirección encontrada es cargada en (CH-ADD) y devuelta en HL.

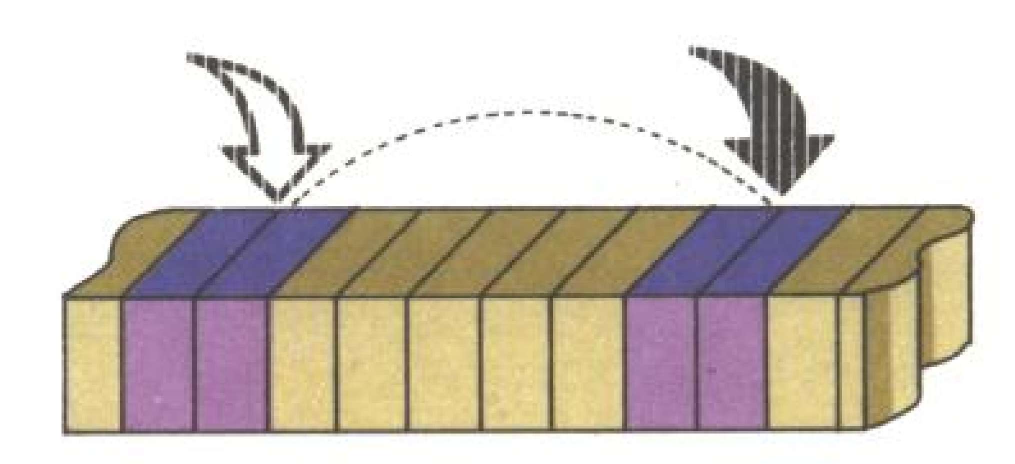
NEXT-ONE 19B8H 6584d

Averigua el comienzo de la próxima línea BASIC o variable y calcula la longitud de la actual.

Datos de entrada: HL: Dirección.
Datos de salida : BC: Longitud de la línea o la variable.
HL: Como entró.
DE: Direc. de la siguiente.

Registros modificados: A,BC,DE.
Variables modificadas: Ninguna.

Rutinas que utiliza: **DIFFER** 19DDH.
Rutina usada por : Múltiples comandos.



DIFFER 19DDH 6621d

Rutina usada por NEXT-ONE y RECLAIM1. Devuelve en BC la diferencia de HL—DE. Intercambia estos registros y hace A = 0.

RECLAIM-1 19E5H 6629d

Elimina la zona de memoria comprendida entre las direcciones señaladas por DE y HL, para ello llama a DIFFER y entra en RECLAIM-2.

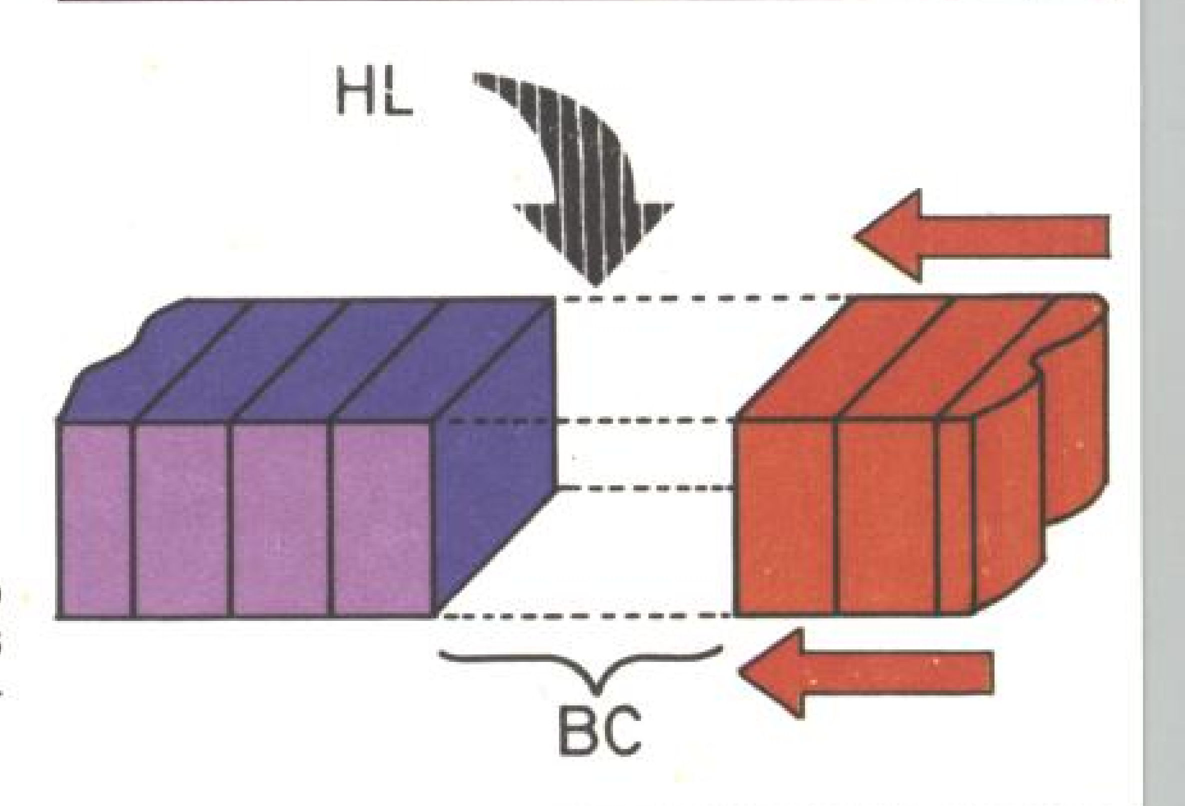
Datos de entrada: DE: Primer byte a borrar.
HL: 1.er byte no borrar.

Resto de datos como RECLAIM-2.

RECLAIM-2 19E8H 6632d

Elimina un bloque de memoria desplazando hacia abajo todo lo que hay tras ella. Todos los punteros del BASIC son actualizados mediante la rutina POINTERS.

Nombre	Hex.	Dec.	
DIFFER	19DDH	6621d	
RECLAIM-1	19E5H	6629d	CIERRA M.
RECLAIM-2	19E8H	6632d	
E-LINE-NO	19FBH	6651d	
OUT-NUM-1	1A1BH	6683d	PRINT NUM
OUT-NUM-2	1A28H	6696d	PRINT NUM



Datos de entrada: HL: Primer byte a borrar.
BC: Longitud por borrar.
Datos de salida : HL: Primer byte de los desplazados.

Registros modificados: A,BC,DE,HL.
Variables modificadas: Los punteros del BASIC.

Rutinas que utiliza: **POINTERS.**
Rutina usada por : Múltiples comandos.

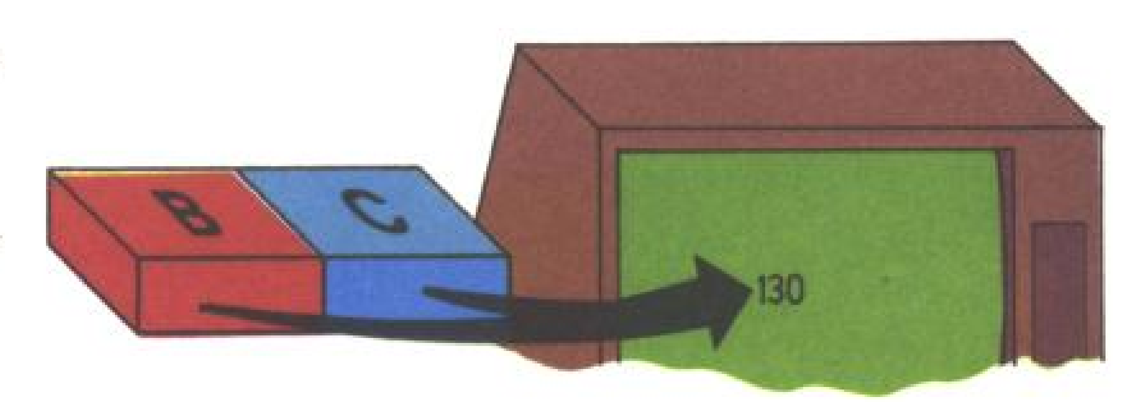
E-LINE-NO 19FBH 6651d

Devuelve en BC el número de línea que se está editando o 0 si no tiene.

OUT-NUM-1 1A1BH 6683d

Escribe el número contenido en el par BC (sólo lo hace correctamente si es menor de 10000).

Datos de entrada: BC.
Datos de salida : Ninguno.



Registros modificados: A,BC y alternativos.
Variables modificadas: Las relativas al canal.

Rutinas que utiliza: **OUT-CODE** 15EFH.
Rutina usada por : **MAIN-5** 133CH.
PRINT-FP2DE3H.

OUT-NUM-2 1A28H 6696d

Igual que OUT-NUM-1 sólo que el número ha de encontrarse en la dirección señalada por HL. Al terminar HL resulta incrementado.

Es usado por OUT-LINE (1855H) para escribir el número de línea.

El bucle de análisis del intérprete BASIC tiene dos entradas:

LINE-SCAN 1B17H 6935d

Es llamada por el bucle principal (MAIN2 12ACH) para chequear la sintaxis de una línea antes de ser incorporada al listado BASIC.

LINE-RUN 1B8AH 7050d

Es llamada por el bucle principal (MAIN3 12CFH) para ejecutar una instrucción o programa.

STMT-LOOP	1B28H	**SCAN-LOOP**	1B52H
GET-PARAM	1B55H	**STMT-RET**	1B76H
LINE-NEW	1B9EH	**LINE-END**	1BB3H
LINE-USE	1BBFH	**NEXT-LINE**	1BD1H
STMT-NEXT	1BF4H	**COM-CLASS**	1C01H

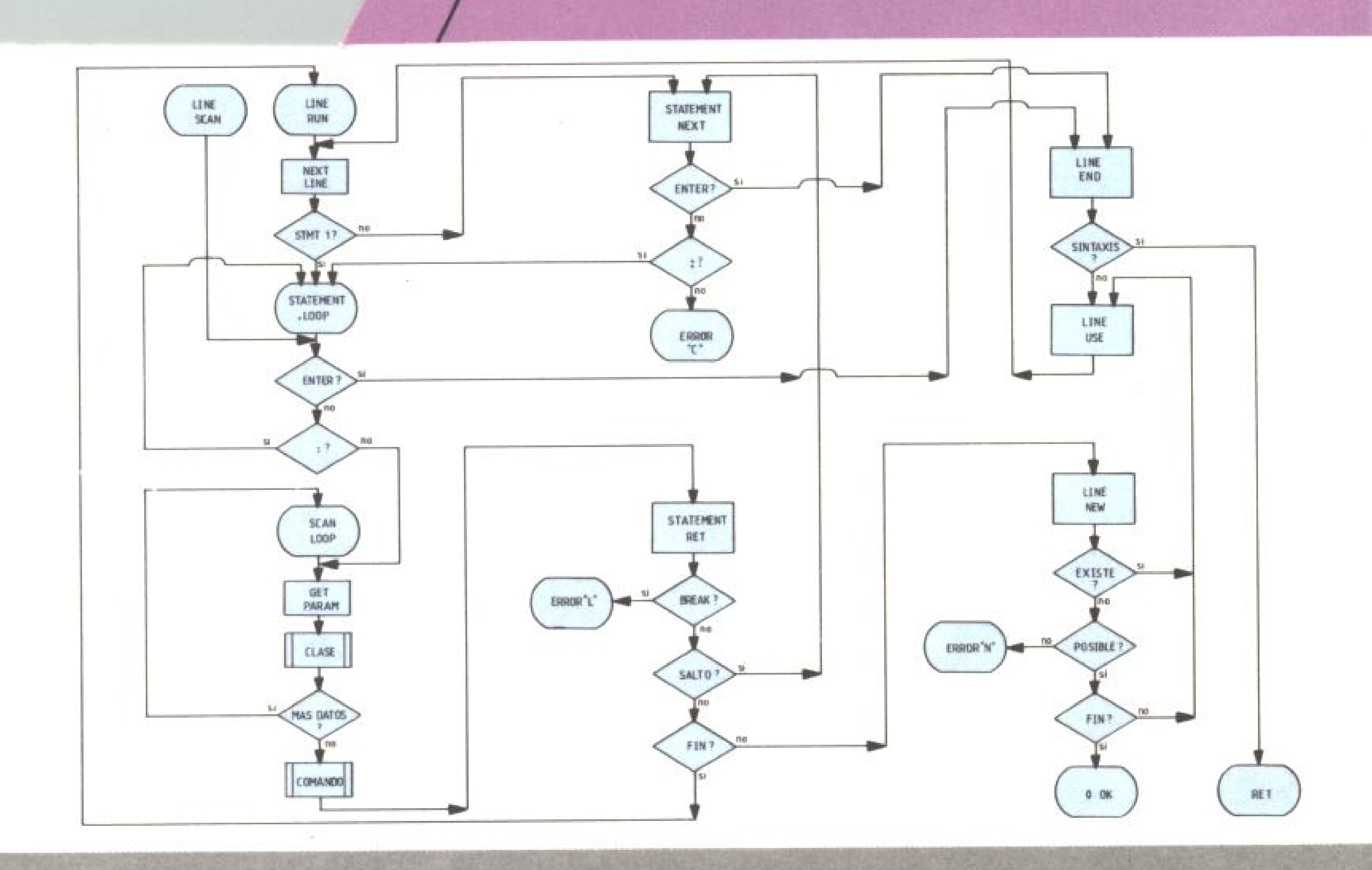
Nombre	**Hex.**	**Dec**
LINE-SCAN	1B17H	6935d
LINE-RUN	1B8AH	7050d
STMT-LOOP	1B28H	6952d

Estas rutinas componen un complejo bucle que se encarga de chequear la sintaxis y ejecutar una a una cada una de las instrucciones que componen el programa.

Para cada comando se ejecutan todas las rutinas de las «clases» que les correspondan (Ver microficha T-8) y, si está chequeando la sintaxis, retorna. En caso contrario, salta a la rutina principal del comando retornando al punto STMT-RET una vez ejecutado.

Tanto la comprobación de la sintaxis como el paso de variables, números y textos a la pila del calculador (STK) es realizado por la rutina **SCANNING** 24FBH (Ver microficha M-34).

En todo el proceso el Bit 7 de la variable FLAGS indica si se está chequeando la sintaxis o ejecutando un comando.



LINE SCAN
LINE RUN
NEXT LINE
STMT 1?
no
si
STATEMENT LOOP
ENTER?
si
no
:?
si
no
SCAN LOOP
GET PARAM
CLASE
MAS DATOS ?
si
no
COMANDO
STATEMENT NEXT
ENTER?
si
no
:?
si
no
ERROR "C"
STATEMENT RET
BREAK ?
si
no
ERROR "L"
SALTO ?
si
no
FIN ?
no
si
LINE END
SINTAXIS ?
si
no
LINE USE
LINE NEW
EXISTE ?
si
no
POSIBLE ?
no
si
ERROR "N"
FIN ?
no
si
0 OK
RET

REM 1BB2H 7090d

Rutina del comando REM. Pasa a la siguiente línea.

VAR-A-1 1C22H 7202d

Esta rutina, a partir de los datos recibidos de LOOK-VARS (28B2H), actualiza las variables STRLEN y DEST o envía el mensaje de error «Variable not found». Es usada por los comandos LET, FOR, NEXT, READ e INPUT.

VAL-FET 1C56H 7254d

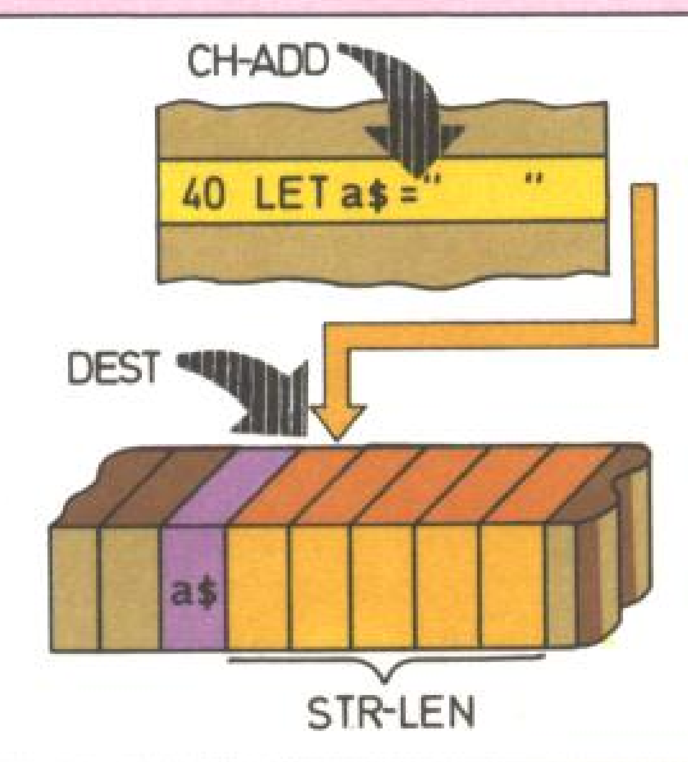
Asigna un valor a la variable BASIC descrita por las variables del sistema STRLEN y DEST.

Datos de entrada: STRLEN, DEST, FLAGS y FLAGX.
CH-ADD señalando al valor.

Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: Múltiples.

Nombre	Hex.	Dec.	
REM	1BB2H	7090d	COMANDO
VAR-A-1	1C22H	7002d	
VAL-FET	1C56H	7254d	
EXPT-2NUM	1C7AH	7290d	→ STK
EXPT-1NUM	1C82H	7298d	→ STK
PERMS	1C96H	7318d	COLOR
FETCH-NUM	1CDEH	7390d	



Rutinas que utiliza: **SCANNING** 24FBH.
LET 2AFFH.
Rutina usada por : Los comandos **LET, READ, INPUT.**

Observaciones: El comando INPUT llama a la rutina a la altura de VAL-FET-2 (1C59H) conteniendo en el acumulador la variable FLAGX.

Si el dato señalado por CH-ADD contiene números, deben estar seguidos de su formato en coma flotante. Para hacer esto puede usarse la rutina SCANNING (24FBH) en modo «chequeo de sintaxis».

EXPT-2NUM 1C7AH 7290d

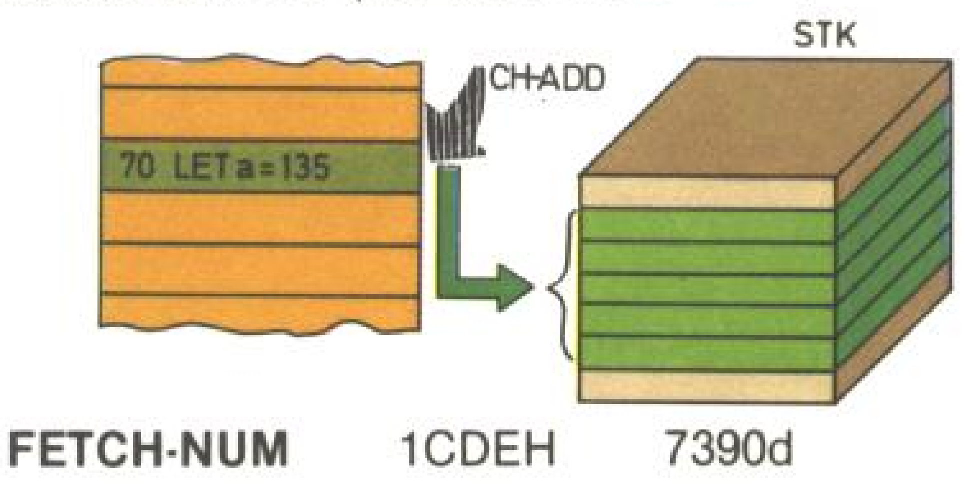
EXPT-1NUM 1C82H 7298d

Lee de la dirección señalada por CH-ADD dos expresiones numéricas separadas por coma, o sólo una, y las guarda en el stack del calculador.
Utilizan la rutina SCANNING (24FBH).

PERMS 1C96H 7318d

Rutina de los 6 comandos de color: INK, PAPER, FLASH, BRIGHT, INVERSE y OVER.

Desde código máquina es más cómodo cambiar directamente las variables del sistema relativas al color (ver microficha G-28).



FETCH-NUM 1CDEH 7390d

Lee de la dirección señalada por CH-ADD una expresión numérica y la guarda en el stack del calculador. En caso de no encontrarla (":" ó ENTER) guarda un 0.

Rutina usada por los comandos clase 3: RANDOMIZE, RESTORE, CLEAR y RUN.
Utiliza la rutina SCANNING (24FBH).

STOP 1CEEH 7406d

Rutina del comando STOP. Produce error 9.

IF 1CF0H 7408d

Rutina del comando IF. Salta a la instrucción o a la línea siguiente, según el resultado de la expresión sea 1 ó 0.

FOR 1D03H 7427d

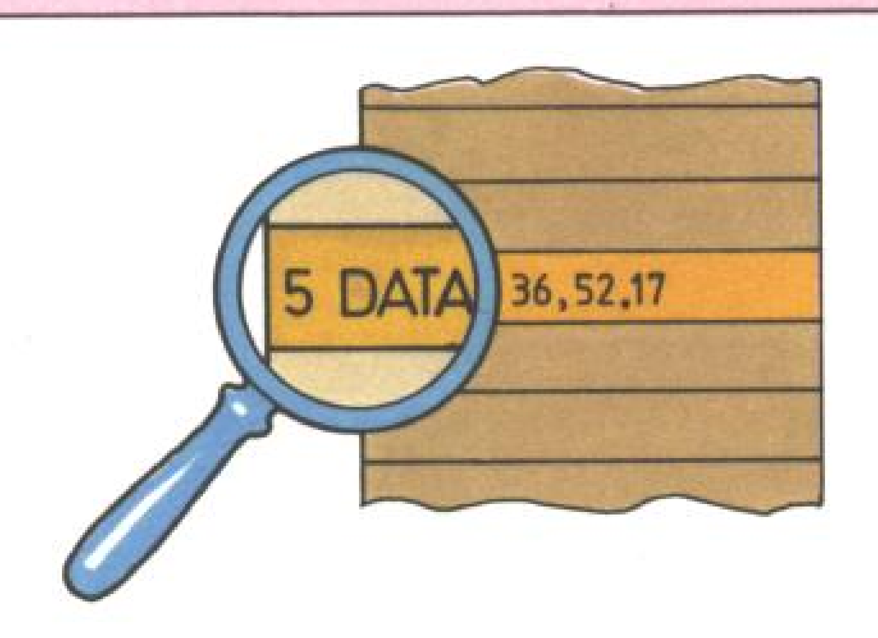
Rutina del comando FOR. Utiliza la rutina LET (2AFFH) y añade tras el valor de la variable los del límite, el salto y el número línea y el de la siguiente instrucción.

LOOK-PROG 1D86H 7558d

Busca un comando en el listado BASIC.

Datos de entrada: HL = Dirección búsqueda.
E = Código del TOKEN.

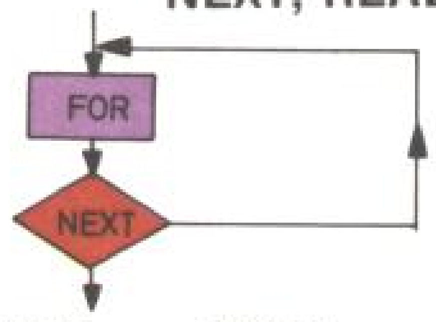
Nombre	Hex.	Dec.	
STOP	1CEEH	7406d	COMANDO
IF	1CF0H	7408d	COMANDO
FOR	1D03H	7427d	COMANDO
LOOK-PROG	1D86H	7558d	
NEXT	1DABH	7595d	COMANDO
READ	1DECH	7660d	COMANDO
DATA	1E27H	7719d	COMANDO
RESTORE	1E42H	7746d	COMANDO
RANDOMIZE	1E4FH	7759d	COMANDO



Datos de salida : BC = Dirección de la línea.
NEWPPC = N.º de línea.
D = Número de instrucción.
HL = CH-ADD = Dirección del TOKEN.
Carry si no fue hallado.

Registros modificados: Múltiples.
Variables modificadas: CH-ADD,NEW-PPC.

Rutinas que utiliza: **EACH-STMT** 198BH.
Rutina usada por : Los comandos **FOR, NEXT, READ, FN.**



NEXT 1DABH 7595d

Rutina del comando NEXT. Incrementa la variable del bucle y salta a la siguiente instrucción o a la siguiente al comando FOR según se haya superado el límite o no.

READ 1DECH 7660d

Rutina del comando READ. Asigna mediante la rutina LET el valor siguiente de la lista DATA.

DATA 1E27H 7719d

Rutina del comando DATA. En modo ejecución salta al próximo comando. En modo sintaxis comprueba los datos y añade el valor en coma flotante.

RESTORE 1E42H 7746d

Rutina del comando RESTORE. Asigna el valor de la variable DATADD.

RANDOMIZE 1E4FH 7759d

Rutina del comando RANDOMIZE. Asigna el valor de la variable SEED. Si es 0 es transferido el valor de los 2 bytes bajos de FRAMES.

CONTINUE 1E5FH 7775d

Rutina del comando CONTINUE. Salta a la instrucción señalada por OLDPPC y OSPPC.

GO-TO 1E67H 7783d

Rutina del comando GOTO. Asigna los valores a las variables NEWPPC y NSPPC.

OUT 1E7AH 7802d

Rutina del comando OUT.

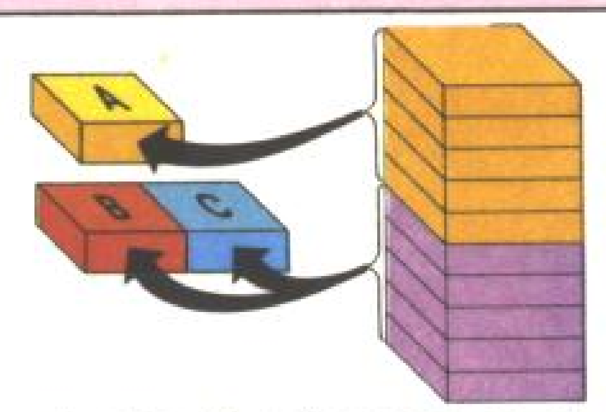
POKE 1E80H 7808d

Rutina del comando POKE.

TWO-PARAM 1E85H 7813

Lee del STACK del calculador un número de un byte complementando a 2 si es negativo (registro A) y un número positivo de 2 bytes (par BC).

Nombre	Hex.	Dec.	
CONTINUE	1E5FH	7775d	COMANDO
GO-TO	1E67H	7783d	COMANDO
OUT	1E7AH	7802d	COMANDO
POKE	1E80H	7808d	COMANDO
TWO-PARAM	1E85H	7813d	←←STK
FIND-INT-1	1E94H	7828d	←STK
FIND-INT-2	1E99H	7833d	←STK
RUN	1EA1H	7841d	COMANDO
CLEAR	1EACH	7852d	COMANDO



Datos de entrada: 2 números en el stack del calculador.

Datos de salida : A = Alto de la pila.
BC = Siguiente dato.

Registros modificados: Múltiples.
Variables modificadas: STK-END.

Rutinas que utiliza: **FP-TO-A** 2DD5H.
FIND-INT-2 1E99H.
Rutina usada por : **OUT** 1E7AH.
POKE 1E80H.

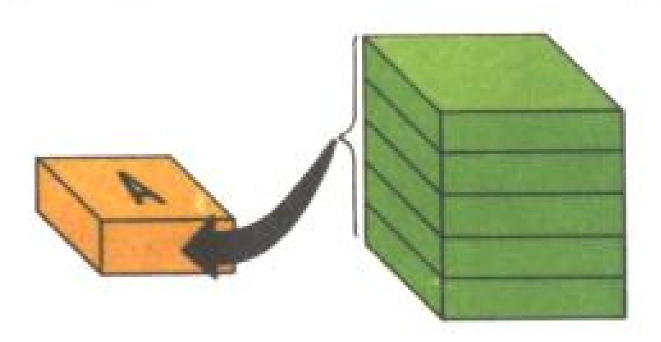
Observaciones: Si los datos exceden de +— 127 o de 65535 se produce error B.

FIND-INT-1 1E94H 7828d

Lee del stack del calculador un número positivo de un byte y lo guarda en el Acumulador.

Si es mayor de 225 o menor que 0 se produce error B.

Utiliza la rutina FP-TO-A (2DD5H).



FIND-INT-2 1E99H 7833d

Lee del stack de calculador un número positivo de dos bytes y lo guarda en el par BC.

Si es mayor de 65535 o menor que 0 se produce error B.

Utiliza la rutina FP-TO-BC (2DA2H).

RUN 1EA1H 7841d

Rutina del comando RUN. Ejecuta las rutinas GOTO, RESTORE 0 y CLEAR.

CLEAR 1EACH 7852d

Rutina del comando CLEAR. Asigna el valor de la variable RAMTOP, llama a CLS (0D6BH) y borra todas las variables.

Para ser utilizado desde CM debe llamarse a la dirección 1EAFH (7855d) teniendo en el par BC la nueva dirección de RAMTOP ó 0.

GO SUB 1EEDH 7917d

Rutina del comando GOSUB. Guarda bajo el stack de máquina la dirección de la instrucción siguiente y llama a la rutina GO-TO.

TEST-ROOM 1F05H 7941d

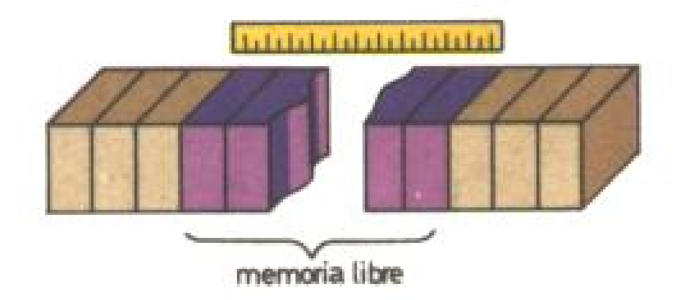
Rutina usada para comprobar si hay suficiente memoria.

Datos de entrada: BC = Bytes que se necesitan.
Datos de salida : HL = Memoria total usada. ERROR 4 si no hay memoria suficiente.

Registros modificados: HL,DE.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **LD-CONTRL** 0808H.
ED-EDIT 0FA9H.
MAKE-ROOM 1655H.
FREE-MEM 1F1AH.

Nombre	Hex.	Dec.	
GOSUB	1EEEH	1917d	COMANDO
TEST-ROOM	1F05H	7941d	
FREE-MEM	1F1AH	7962d	
RETURN	1F23H	7971d	COMANDO
PAUSE	1F3AH	7994d	COMANDO
PAUSE-1	1F3DH	7997d	



FREE-MEM 1F1AH 7962d

En Basic no existe el comando FREE pero puede implementarse mediante PRINT 65536-USR 7962.

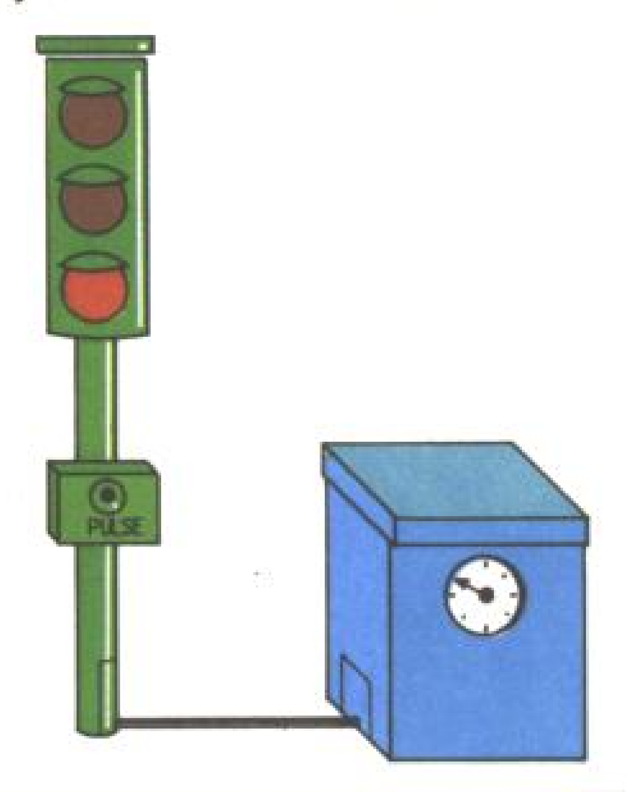
Esta rutina llama a TEST-ROOM con 0 en el par BC y posteriormente transfiere a BC el valor del par HL (memoria ocupada).

RETURN 1F23H 7971d

Rutina del comando RETURN. Lee debajo del stack de máquina la dirección de retorno y salta a la rutina GO-TO.

PAUSE 1F3AH 7994d

Rutina del comando PAUSE. Lee del STACK un número y entra en PAUSE-1.



PAUSE-1 1F3DH 7991d

Espera durante el tiempo indicado por el par BC en 1/50 de segundo o hasta que sea pulsada una tecla.

Datos de entrada: BX = Tiempo (0 significa infinito).
Datos de salida : BC = A = 0.
RES 5 (FLAGS).

Registros modificados: A,BC.
Variables modificadas: BIT 5 (FLAGS).

Rutinas que utiliza: Interrupciones enmascarables.
Rutina usada por : El comando PAUSE.

Observaciones: Para el funcionamiento de esta rutina deben estar habilitadas las interrupciones (EI).

Para anular la pulsación de tecla anterior debe hacerse RES 5, (FLAGS).

BREAK-KEY 1F54H 8020d

Comprueba si fue pulsado BREAK.

Datos de entrada: Ninguno.
Datos de salida : Carry si no se pulsó.

Registros modificados: AF.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **COPY** 0EACH.
STMT-RET 1B76H.

Observaciones: Funciona aunque estén deshabilitadas las interrupciones.

Para incorporar el comando BREAK a un programa en código máquina debe colocarse en el bucle principal la siguiente rutina:

```
CALL   BREAK-KEY;     1F54H
JP     NC,ERROR-L;    1B7BH
```

O cualquier otra que restaure el STACK.

Nombre	Hex.	Dec.	
BREAK-KEY	1F54H	8020d	BREAK
DEF-FN	1F60H	8032d	
UNSTACK-Z	1FC3H	8131d	
LPRINT	1FC9H	8137d	COMANDO
PRINT	1FCDH	8141d	COMANDO
PRINT-2	1FDFH	8159d	
INPUT	2089H	8329d	COMANDO
IN-CHAN-K	21D6H	8662d	
CO-TEMP	21E1H	8673d	
BORDER	2294H	8852d	COMANDO

DEF-FN 1F60H 8032d

Rutina del comando DEF FN. En modo ejecución salta al próximo comando. En modo sintaxis comprueba los datos y abre los espacios necesarios para que FN guarde los parámetros (ver microficha G-26).

UNSTACK-Z 1FC3H 8131d

Rutina usada por casi todos los comandos. Si se está chequeando la sintaxis «BIT 7, (FLAGS)» no retorna a donde fue llamada sino a la dirección anterior (normalmente STMT-RET 1B76H). Si está en modo ejecución retorna a donde fue llamada.

LPRINT 1FC9H 8137d

Rutina del comando LPRINT.

PRINT 1FCDH 8141d

Rutina del comando PRINT.

PRINT-2 1FDFH 8159d

Parte común de LPRINT,PRINT e INPUT.

INPUT 2089H 8329d

Rutina del comando INPUT.

Utiliza PRINT-2 (1FDFH), EDITOR (0F2CH) y LET (2AFF) directamente o a través de VAL-FET (1C56H).

IN-CHAN-K 21D6H 8662d

Test de utilización del canal K. Pone a cero la bandera Z si se está utilizando un canal marcado con la letra K. Utiliza el par de registros HL.

CO-TEMP 21E1H 8673d

Rutina de control de los comandos de color.

BORDER 2294H 8852d

Rutina del comando BORDER. Cambia el color del borde y asigna el color de tinta que más contraste (blanco o negro).

Puede ser llamada desde código máquina con el código de color en el stack del calculador.

También puede ser llamada en la dirección 2297H (8855d) con el número de color en el acumulador).

Utiliza solamente el registro A y cambia el valor de la variable del sistema (BORDCR).

PIXEL-ADD 22AAH 8874d

Calcula la dirección de un pixel en el archivo de imagen.

Datos de entrada: BC = Coordenadas (B = y C = x).
Datos de salida : HL = Dirección.
A = N.º de bit en el byte.

Registros modificados: AF,B,HL.
Variables modificadas: Ninguna.

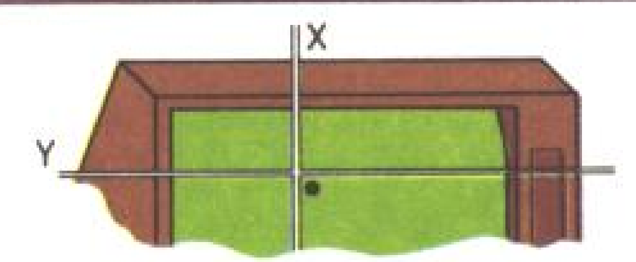
Rutinas que utiliza: Ninguna.
Rutina usada por : **POINT** (22CBH).
PLOT (22DCH).

POINT-SUB 22CBH 8907d STK
22CEH 8910D BC

Rutina del comando POINT. Comprueba el estado de un bit en el archivo de imagen.

Datos de entrada: STK numérico = dirección.
Datos de salida : STK numérico = 1 ó 0.

Nombre	Hex.	Dec.	
PIXEL-ADD	22AAH	8874d	
POINT-SUB	22CBH	8907d	COMANDO
POINT-BC	22CEH	8910d	
PLOT	22DCH	8924d	COMANDO
PLOT-BC	22DFH	8927d	
STK-TO-BC	2307H	8967d	←←STK



Registros modificados: Múltiples.
Variables modificadas: Ninguna.

Rutinas que utiliza: **STK-TO-BC** 2307H.
PIXEL-ADD 22AAH.
STACK-A 2D28H.
Rutina usada por : El comando POINT.

Observaciones: Puede ser llamada a la dirección 22CEH con la dirección del punto en el par BC.

PLOT 22DCH 8924d STK
22DFH 8927d BC

Rutina del comando PLOT. Dibuja o borra un punto en las coordenadas indicadas.

Datos de entrada: Dirección en el STACK numérico.
PFLAG indicando OVER o INVERSE.
Datos de salida : Ninguno.

Registros modificados: Múltiples.
Variables modificadas: COORDS.

Rutinas que utiliza: **STK-TO-BC** 2307H.
PIXEL-ADD 22AAH.
PO-ATTR 0BDBH.
TEMPS 0D4DH.
Rutina usada por : **CIRCLE** 2320H
DRAW 2382H.

Observaciones: Puede ser llamada a la dirección 22DFH con la dirección del punto en el par BC.

Para establecer los colores temporales puede llamarse a la rutina TEMPS (0D4DH) con el bit 0 de TV-FLAG puesto a 0.

STK-TO-BC 2307H 8967d

Obtiene del stack del calculador dos números enteros entre —255 y +255. Su valor absoluto es cargado en el par BC y sus signos (+—1) en el par DE.
Datos de entrada: 2 números en el STACK numérico.
Datos de salida : B número. D Signo.
C número. E signo (A = C).

Registros modificados: Múltiples.
Variables modificadas: STK-END.

Rutinas que utiliza: **STK-TO-A** 2314H.
Rutina usada por : Múltiples comandos.

Observaciones: Los registros B y D se corresponden con el valor de lo alto de la pila y C y E con los del siguiente.

Comandos VIII - Circle, Draw

CIRCLE 2320H 8992d

Rutina del comando CIRCLE. Dibuja una circunferencia e torno a un punto dado.

CIRCLE-1 232DH 9005d

Continuación de la rutina circle. Punto de entrada para la utilización de la rutina desde código máquina.

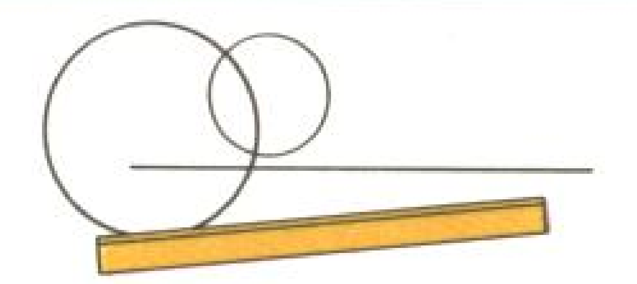
Datos de entrada: x, y,radio en el STACK del calculador.
Datos de salida : Ninguno.

Registros modificados: Múltiples. (Incluso HL')
Variables modificadas: COORDS y STK-END

Rutinas que utiliza: **PLOT** 22DCH.
DRAW 2382H.
FP-CALC 0028H.
Rutina usada por : El comando CIRCLE.

Observaciones: Ver las correspondientes a DRAW.

Nombre	**Hex.**	**Dec.**	
CIRCLE	2320H	8992d	COMANDO
CIRCLE-1	232DH	9005d	Circunf.
DRAW	2382H	9090d	COMANDO
DR3-PRMS1	2394H	9108d	Curva
LINE-DRAW	2477H	9335d	Recta
DRAW-LINE	24B7H	9399d	Recta
DRAW-LINE-1	24BAH	9402d	Recta



DRAW 2382H 9090d

Rutina del comando DRAW. Puede ser llamada desde código máquina a diferentes puntos:

Línea curva (Tres parámetros).

DR-3-PRMS-1 238DH 9108d
x,y,ángulo en el Stack del calculador.

Línea recta (sólo dos parámetros):

LINE-DRAW 2477H 9335d

Dos números en el stack del calculador (x,y). No llama a TEMPS.

DRAW-LINE-1 24BAH

B = y, C = x, D = signo de B (+ —1), E = signo de C (+ —1).

Al finalizar la rutina es conveniente llamar a TEMPS (0D4DH) para restablecer los colores permanentes.

En todos los casos:

Datos de entrada: Ninguno.

Registros modificados: Múltiples.
Variables modificadas: COORDS, STK-END.

Rutinas que utiliza: **FP-CALC** 0028H.
PLOT 22DCH.
TEMPS 0D4DH (no todos).
Rutina usada por : El comando CIRCLE.

Observaciones: Para que el dibujo se haga en los colores que se deseen, éstos deben estar en las variables de color temporales. Para conseguir esto puede llamarse a la rutina TEMPS (0D4DH) con el BIT 0 de TV-FLAG a 0.

- Estas rutinas alteran el registro HL' por lo que debe restablecerse su valor (2758H = 10072d) antes de volver al BASIC.

Ejemplo:

```
RES     0,TV-FLAG
CALL    TEMPS
LD      B,desp y
LD      D,signo desp y
LD      C,desp x
LD      E,signo desp x
CALL    DRAW-LINE-1
LD      HL,10072
EXX
```

SCANNING 24FBH 9467d

Esta es la más compleja de las rutinas de la ROM. Tiene dos modos de funcionamiento según indique el bit 7 de la variable FLAGS (IY + 1).

En modo «sintaxis», RES 7, (FLAGS); Comprueba la correcta colocación de los operandos, paréntesis, etc. de las expresiones e intercala después de cada número su valor en coma flotante.

En modo funcionamiento, «run», SET 7, (FLAGS); evalúa una expresión guardando su valor si es numérica o sus parámetros si es alfanumérica en el stack del calculador. Cuando la expresión es compleja guarda todos los valores y efectúa las operaciones necesarias. Para ello tiene en cuenta todas las funciones y la tabla de prioridades.

Datos de entrada: CH-ADD apuntando a la expresión.

Nombre	**Hex.**	**Dec.**	
SCANNING	24FBH	9467d	→ STK
S-SCRN$-S	2535H	9525d	FUNCION
S-SCRNS-1	253FH	9535d	
S-ATTR-S	2580H	9600d	FUNCION

Datos de salida :
- BIT 6, (FLAGS) = 1 si es numérico.
  Valor en lo alto de la pila.
- BIT 6, (FLAGS) = 0 si es alfanumérico.
- En lo alto de la pila:
  1.er byte indeterminado.
  2.º y 3.er bytes dirección.
  4.º y 5.º bytes longitud.

Registros modificados: Múltiples.
Variables modificadas: Múltiples.

Rutinas que utiliza: Múltiples, incluso a sí misma recursivamente.
Rutina usada por : Múltiples rutinas.

S-SCRN$-S 2535H 9525d

Rutina de la función SCREEN$. A partir de dos datos en el stack del calculador indicado línea y columna devuelve en el mismo stack los parámetros de una cadena vacía o un carácter creado en el espacio de trabajo con un código igual al encontrado en la dirección de pantalla indicado.

Datos de entrada: Línea y Columna en el stack del calculador.
CHARS señalando a tabla caracteres-256.
Datos de salida : Parámetros alfanuméricos en el stack del calculador.

S-CRN$-1 253FH 9535d

Es continuación de la rutina anterior puede llamarse en las siguientes condiciones:

Datos de entrada: C = Línea (0-23).
B = Columna (0-31).
HL = Dirección carácter 32.

S-ATTR-S 2580H 9600d

Rutina de la función ATTR. A partir de dos datos en el stack del calculador indicando línea y columna devuelve, en el mismo stack, el código de los colores que constituyen los atributos del carácter allí situado.

Datos de entrada: Línea y Columna en el stack del calculador.
Datos de salida : Código de los atributos en el stack del calculador:
128 * FLASH + 64 * BRILLO + 8 * PAPEL + TINTA.

Registros modificados: Múltiples.
Variables modificadas: STKEND.

Rutinas que utiliza: **STK-TO-BC** 2307H.
STACK-A 2D28H.
Rutina usada por : La función ATTR.

Observaciones: Esta rutina puede ser llamada a la dirección 2583H (9603d) con el número de línea en C y el de columna en B.

LOOK-VARS 28B2H 10418d

Busca una variable en el área de variables BASIC o en la zona de los argumentos de un comando DEF-FN si DEFADD no contiene 0.

Datos de entrada: CH-ADD señalando al nombre de la variable.
DEFADO = 0 o señalando a DEF-FN.

Datos de salida :
- Variable no encontrada:
Bandera de Carr = 1 (C).
Z si era un array.
HL señala primer carácter en el área del listado.
- Variable encontrada:
Bandera de Carry = 0 (NC).
Z cadena simple o cualquier array.
HL señala al último carácter del nombre en el área de variables.

Nombre	**Hex.**	**Dec.**
LOOK-VARS	28B2H	10418d
STK-VAR	2996H	10646d→STK
SLICING	2A52H	10834d→STK

- En todos los casos:
Bits 5 y 6 de C = Tipo.
00: Matriz numérica.
01: Numérica varias letras.
10: Alfanumérica.
11: Numérica una letra.
Bit 7 complemento del bit 7 de FLAGS (1 = sintax 0 = ejec.).
Bits 0 a 4 Código del nombre 1 = > A , 2 = > B , etc.

Registros modificados: Múltiples.
Variables modificadas: Múltiples.

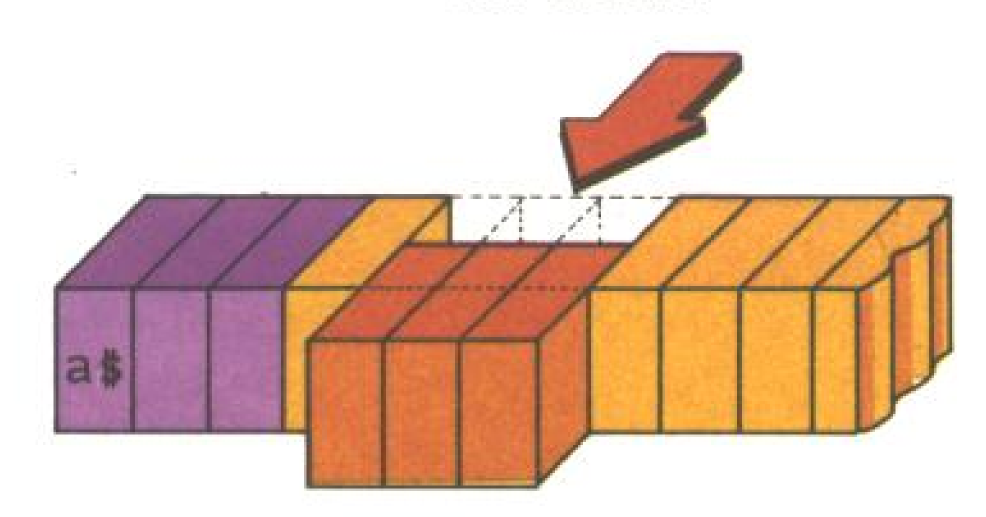
Rutinas que utiliza: **GET-CHAR** 0018H.
NEXT-CHAR 0020H.
NEXT-ONE 19B8H.
ALPHA 2C8DH.
ALPHANUM 2C88H.
Rutina usada por : **SAVE-ETC** 0605H.
CLASS-1 1C1FH.
CLASS-4 1C6CH.
SCANNING 24FBH.
DIM 2C02H.

STK-VAR 2996H 10646d

Esta rutina se encarga de guardar en el stack el valor de una variable numérica, los parámetros de un string o un elemento de un array tanto numérico como alfanumérico.

Datos de entrada: Los de salida de LOOK-VARS.
Datos de salida : En el stack del calculador.
Registros modificados: Múltiples.
Variables modificadas: CH-ADD.

Rutinas que utiliza: **GET-CHAR** 0018H.
SLICING 2A52H.
STK-STORE 2AB2H.
GET-HL * DE 2AF4H.
Rutina usada por : **VAR-A-2** 1C30H.
SCANNING 24FBH.
DIM 2C02H.

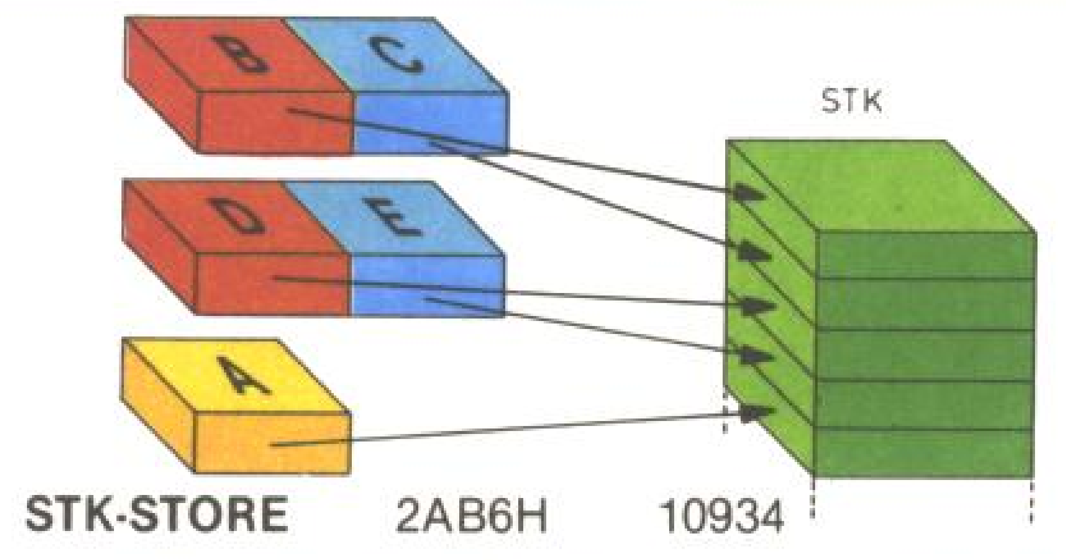


SLICING 2A52H 10834d

Rutina que corta las variables alfanuméricas en las expresiones tipo (n TO m).

Es usada por SCANNING (24FBH) y STK-VAR (2996H).





STK-STORE 2AB6H 10934

Guarda en el stack del calculador un número o los parámetros de una variable contenidos en los registros A,E,D,C,B, por este orden.

Datos de entrada: — Si es una cadena:
DE = comienzo
BC = longitud.
— Si es un número:
A = mantisa,
EDCB = Argumento.

Datos de salida : Dato en el stack del calculador.
HL = Nuevo STKEND.

Nombre	Hex.	Dec.
STK-STORE	24FBH	10934d→STK
INT-EXP	2ACCH	10956d
DE,(DE + 1)	2AEEH	10990d
LET	2AFFH	11007d COMANDO
L-ENTER	2BA6H	11174d
STK-FETCH	2BF1H	11249d←STK

Registros modificados: HL.
Variables modificadas: STKEND.

Rutinas que utiliza: **TEST-5-SP** (33A9H).
Rutina usada por : Múltiples comandos.

Observaciones: La función inversa es realizada por la rutina STK-FETCH (2BF1H).

Hay dos entradas a la rutina aparte de esta:

—STK-ST-0 (2AB1H) que hace XOR A y RES 6,(FLAGS) para indicar que se almacena una parte de una variable alfanumérica.

—STK-STO-$ (2AB2H) que hace RES 6,(FLAGS) para indicar que se almacena una variable alfanumérica.

INT-EXP 2ACCH 10956d

Sitúa en el par de registros BC el resultado de la próxima expresión (señalada por CH-ADD) en forma de un entero. Si hay desbordamiento el carry es puesto a 1 y A contiene FFH.

DE,(DE + 1) 1AEEH 10990d

Carga en el par DE el valor señalado por DE + 1.

Retorna con HL señalando a DE + 2 (se entiende el valor inicial de DE). Utiliza HL y DE.

LET 2AFFH 11007d

Asigna el valor situado en lo alto del STACK a la variable descrita por DEST y STRLEN.

Es usada por LET,READ e INPUT.

L-ENTER 2BA6H 11174d

Intercambia los valores de HL y DE y retorna si el par BC contiene 0. En caso contrario hace un LDIR y retorna recuperando el valor inicial de HL, con A, B y C = 0 y DE = DE + BC.

STK-FETCH 2BF1H 11249d

Lee un dato del stack numérico cargándolo en los registros A,E,D,C,B ajustando el nuevo valor de STKEND, que al mismo tiempo es devuelto en el par de registros HL.

Datos de entrada: En el stack del calculador.
Datos de salida : — Si es una cadena:
DE = comienzo
BC-longitud.
— Si es un número:
A = mantisa,
EDCB = Argumento.
— En ambos casos HL = nuevo STKEND.

Registros modificados: AF,BC,DE,HL.
Variables modificadas: STKEND.

Rutinas que utiliza: Ninguna.
Rutina usada por : Múltiples comandos.

Observaciones: Es la rutina inversa de STK-STORE (2AB6H).

DIM 2C02H 11266d

Rutina del comando DIM. Abre un espacio en la zona de variables y lo formatea.

ALPHANUM 2C88H 11400d

Retorna con el flag de carry a 1 si el valor contenido en el acumulador corresponde a una letra o un dígito. Modifica sólo el registro F.

ALPHA 2C8DH 11405d

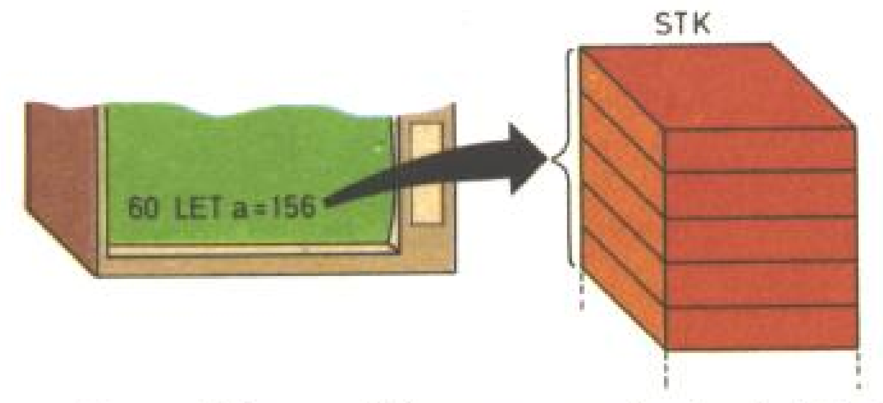
Retorna con el flag de carry a 1 si el valor contenido en el acumulador corresponde a una letra. Modifica solamente el registro F.

DEC-TO-FP 2C9BH 11419d

Guarda en el stack del calculador un número en código ASCII en cualquiera de los tres formatos (**BIN**ario, decimal o **E**xponencial).

Datos de entrada: CH-ADD señalando al número.
A = Primera cifra.

Nombre	Hex.	Dec.
DIM	2C02H	11266d COMANDO
ALPHANUM	2C88H	11400d
ALPHA	2C8DH	11405d
DEC-TO-FP	2C9BH	11419d→STK
NUMERIC	2D1BH	11547d
STK-DIGIT	2D22H	11554d→STK
STACK-A	2D28H	11560d→STK
STACK-BC	2D2BH	11563d→STK



Datos de salida : Número en el stack del calc.
HL = nuevo CH-ADD.

Registros modificados: Múltiples.
Variables modificadas: CH-ADD, STKEND.

Rutinas que utiliza: Múltiples.
Rutina usada por : **SCANNING** 24FBH.

Observaciones: Si el primer carácter no es un número ni «BIN» guarda un 0.

NUMERIC 2D1BH 11547d

Retorna con el flag de carry a 1 si el valor contenido en el acumulador corresponde a un dígito.

Modifica solamente el registro F.

STK-DIGIT 2D22H 11554d

Guarda en el stack del calculador el valor del dígito contenido en el registro A en código ASCII.

Si no corresponde a ningún dígito retorna con el flag de carry alzado y ningún registro alterado salvo F.

Si corresponde a un dígito resta 30 al acumulador y entra en STACK-A.

STACK-A 2D28H 11560d

Guarda en el stack del calculador el valor contenido en el acumulador.

Guarda A en BC y entra en STACK-BC.

STACK-BC 2D2BH 11563d

Guarda en el stack del calculador el valor contenido en el par de registros BC.

Datos de entrada: BC = número por guardar.
Datos de salida : Número en el stack del calc.
HL = Antiguo STKEND (número).
DE = Nuevo STKEND.
Carry flag a 0 (NC).

Registros modificados: Múltiples.
Variables modificadas: STK-END.

Rutinas que utiliza: **STK-STORE** 2AB6H.
FP-CALC 0028H.
Rutina usada por : Múltiples comandos.

INT-TO-FP 2D3BH 11579d

Guarda en el stack del calculador un número natural en código ASCII.

Datos de entrada: A = Primer carácter.
CH-ADD apuntando a éste.
Datos de salida : Número en el stack del calc.
CH-ADD apuntando al siguiente carácter.

Registros modificados: Múltiples.
Variables modificadas: STKEND, CH-ADD.

Rutinas que utiliza: **FP-CALC** 0028H.
STK-DIGIT 2D22H.
CH-ADD + 1 0074H.
Rutina usada por : **E-LINE-NO** 19FBH.
DEC-TO-FP 2C9BH.

Observaciones: Si el primer carácter no es un dígito guarda un 0.

Nombre	Hex.	Dec.
INT-TO-FP	2D3BH	11579d →STK
INT-FETCH	2D7FH	11647d ←STK
P-INT-STO	2D8CH	11660d →STK
INT-STORE	2D8EH	11662d →STK

INT-FETCH 2D7FH 11647d

Lee de la dirección señalada por el par HL un pequeño entero (—65535 < = n < = 65535).

Esta dirección suele encontrarse en el stack del calculador.

Datos de entrada: HL = Dirección.
Datos de salida: : DE = Valor absoluto.
C = Signo (0 pos. —1 neg.)
HL incrementado en 3.
A = D.

Registros modificados: AF,C,DE,HL.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **FP-TO-BC** 2DA2H y otras.

Observaciones: Esta rutina no elimina el número contenido en el stack del calculador.
Su rutina inversa es INT-STORE (2D8EH).

P-INT-STO 2D8C 11660d

Almacena un pequeño número natural (0 > = n < = 65535). Carga en C un 0 y entra en INT-STORE.

INT-STORE 2D8EH 11662d

Almacena en la dirección señalada por el par HL un pequeño entero (—65535 < = n < = 65535).
Esta dirección suele encontrarse en el stack del calculador.

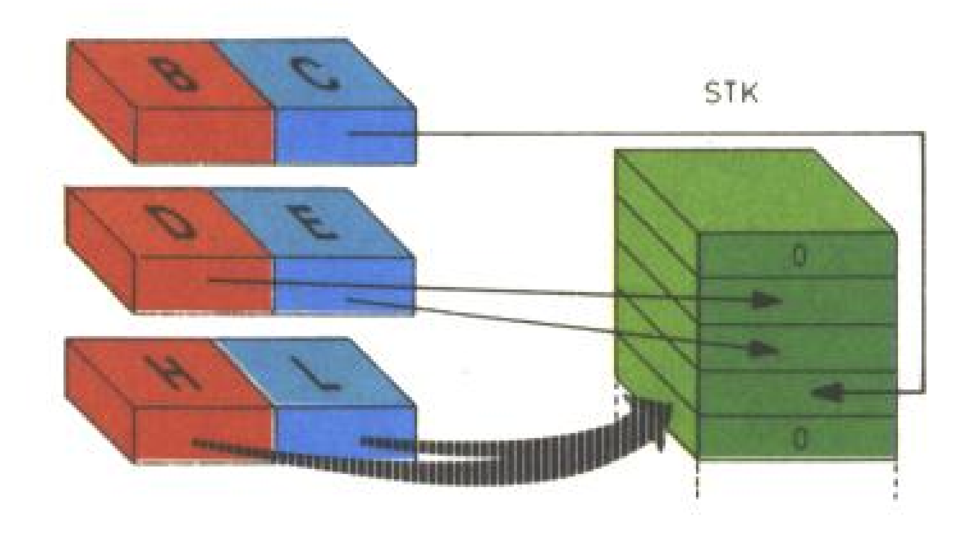
Datos de entrada: HL = Dirección.
DE = Valor absoluto.
C = Signo (0 pos. —1 neg.).

Datos de salida : HL como entró.

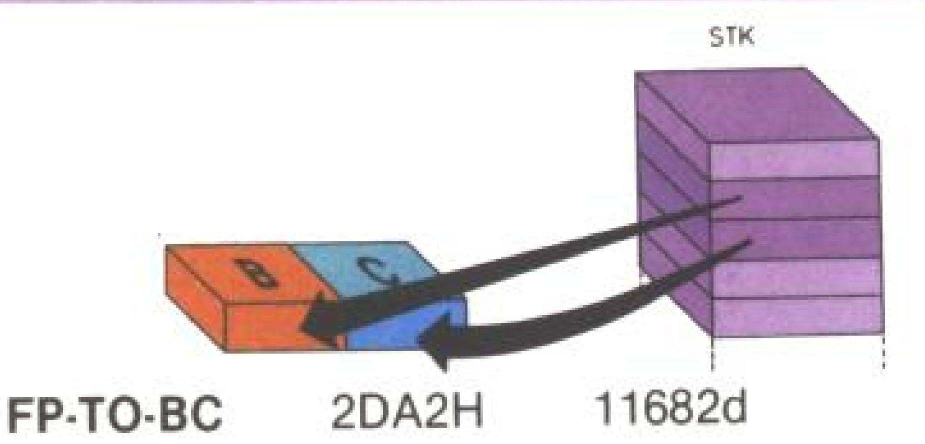
Registros modificados: AF.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : Múltiples comandos.

Observaciones: Esta rutina no actualiza la variable STK-END por lo que no se añade al stack del calculador.
Su rutina inversa es INT-FETCH (2D7FH).







FP-TO-BC 2DA2H 11682d

Lee del stack del calculador un pequeño número en complemento a 2 (—65535 a 65535) aproximado a la parte entera.

Datos de entrada: Número en el stack del calc.
Datos de salida : BC = valor absoluto.
A = C.
Flag Z si es positivo (NZ si es neg.)
Carry si hay exceso (es mayor de 65535.5 o menor de —65535.5.
HL = Nuevo STKEND—5 (siguiente número).
DE = Nuevo STKEND. (número obtenido).

Nombre	Hex.	Dec.
FP-TO-BC	2DA2H	11682d ←STK
FP-DELETE	2DADH	11693d ←STK
FP-TO-A	2DD5H	11733d ←STK
PRINT-FP	2DE3H	11747d P. NUMERO
CA = 10 * A + C	2F88H	12171d
HL = HL * DE	2DA9H	12457d

Registros modificados: Múltiples.
Variables modificadas: STK-END.

Rutinas que utiliza: **FP-CALC** 0028H.
IN-FETCH 2D7FH.

Rutina usada por : **E-LINE-NO** 19FBH.
FIND-INT-2 1E99H.
SCANNING 24FBH.
FP-TO-A 2DD6H.

Observaciones: Esta rutina es la que utiliza FIND-INT-2 (1E99H) produciendo aquélla un mensaje de error si retorna con NZ o Carry. Si no se desea esto debe usarse FP-TO-BC.

FP-DELETE 2DADH 11693d

Lee del stack del calculador la parte entera de un pequeño número en complemento a 2 (—65535 a 65535). Se diferencia de FP-TO-BC cuando la parte decimal es mayor de 0.5. Ej: si el número es 8.6 FP-TO-BC nos devolvería 9 y FP-DELETE 8.

FP-TO-A 2DD5H 11733d

Lee del stack del calculador un pequeño número en complemento a 2 (—255 a 255) aproximado a la parte entera.

Todas las condiciones son como FP-TO-BC excepto en que el flag de carry se pone a 1 cuando el número es mayor de 255.5 o menor de —255.5.

PRINT-FP 2DE3H 11747d

Escribe el número contenido en lo alto del stack del calculador en el canal actual (abierto con CHAN-OPEN 1601H).

Si el número es excesivamente grande o pequeño es escrito en el formato exponencial.

Los punteros del canal correspondiente son actualizados y el número eliminado del stack.

Es utilizado por el comando PRINT (1FCFH) y por la función STR$ (361FH).

CA = 10 * A + C 2F8BH 12171d

Rutina usada por PRINT-FP. Calcula en HL 10 * A + C y posteriormente transfiere H a C y L a A.

Modifica solamente estos 4 registros.

HL = HL * DE 30A9H 12457d

Efectúa una multiplicación de 16 bits.

Datos de entrada: HL,DE.
Datos de salida : HL = Anterior HL * DE.

Registros modificados: HL,AF.
Variables modificadas: Ninguna.

Rutinas que utiliza: Ninguna.
Rutina usada por : **GET HL * DE** 2AF4H.
multiply 30CAH.

STACK-NUM 33B4H 13236d

Transfiere un número en formato de coma flotante al stack del calculador.

Datos de entrada: HL = Dirección.
Datos de salida : DE = Nuevo STKEND.
HL = →Detrás del número.
BC = 0.

Registros modificados: BC,DE,HL.
Variables modificadas: STKEND.

Rutinas que utiliza: **TEST-ROOM** 1F05H.
Rutina usada por : **BEEP** 03F8H.
SCANNING 24FBH.

SWAP-BYTE 334EH 13374d

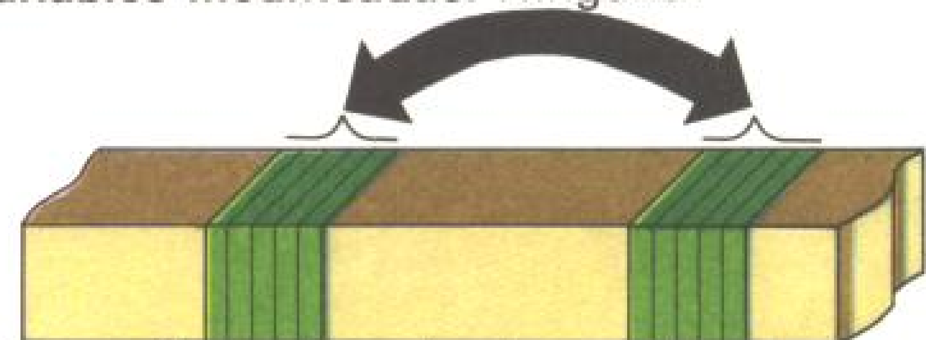
Intercambia los contenidos de las zonas de memoria señalados por los pares de registros HL y DE de una longitud determinada por el registro B.

Datos de entrada: HL y DE = Punteros.
B = Longitud bloques.

Nombre	Hex.	Dec.
STACK-NUM	33B4H	13236d →STK
SWAP-BYTE	343EH	13374d
TEST-ZERO	34E9H	13545d
STK-PNTRS	35BFH	13759d
SP-SPACE	386EH	14446d
CHARS-T	3D00H	15616d TABLA

Datos de salida : Ninguno.

Registros modificados: AF,BC,DE,HL.
Variables modificadas: Ninguna.



Observaciones: La entrada «exchange» carga en B el valor 5 y entra en SWAP-BYTE. Al término HL contiene anterior DE +5 y DE anterior HL +5.

TEST-ZERO 34E9H 13545d

Mira si 4 bytes señalados por el par HL contienen 0.

Datos de entrada: HL señalando al primer byte.
Datos de salida : Carry flag y Z si los 4 bytes son 0.

Registros modificados: F.
Variables modificadas: Ninguna.

STK-PNTRS 35BFH 13759d

Sitúa HL apuntando al primer byte del número que se encuentra en lo alto del stack del calculador y DE encima de la pila.

Datos de entrada: Ninguno.
Datos de salida : HL = STKEND — 5.
DE = STKEND.

Registros modificados: Ninguno.
Variables modificadas: Ninguna.

Espacio de separación 386EH 14446d

Entre las direcciones 386EH y 3CFFH (15615d) se encuentran algo más de 1K (1170 bytes) que contienen FFH (Todos los bits a 1).

Esta zona es el espacio que sobró al hacer la ROM, pero tiene gran utilidad pues aquí pueden situarse mediante hardware ciertas rutinas de algunos periféricos que han de ser compatibles con la ROM.

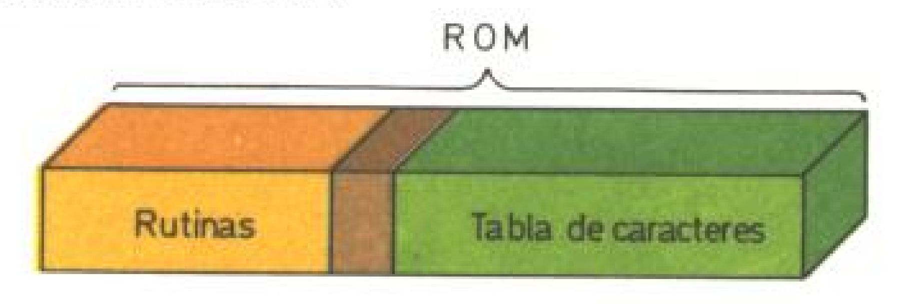


Tabla de caracteres 3D00H 15616H

En los últimos 768 bytes se encuentran las tablas de los 96 gráficos ordinarios.

Esta dirección es la señalada inicialmente por la variable del sistema CHARS (5C36H, 23606d) pero puede ser cambiada a voluntad para crear todos los nuevos caracteres que se deseen.

CALCULATE 335BH 13147d

Rutina del del calculador. Sirve tanto para hacer cálculos numéricos como alfanuméricos.

Después de la llamada se sitúan una serie de bytes que indican las operaciones a realizar, debiendo terminar en el código 38H que determina el fin de los cálculos.

Stack del calculador (STK)

La zona de memoria situada entre las direcciones señaladas por los punteros STKBOT y STKEND constituye el stack o pila del calculador. Su misión es el almacenamiento temporal de datos para hacer las operaciones siguiendo las reglas de prioridad.

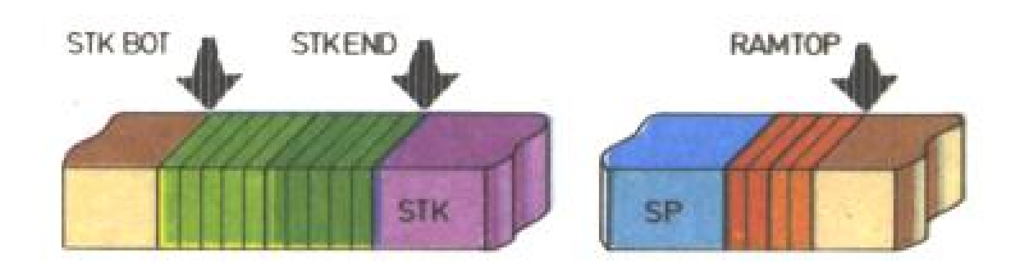
Esta pila crece al revés que el stack o pila de máquina, pues mientras los datos de ésta se almacenan hacia las partes bajas de la memoria, los datos del calculador se almacenan de abajo hacia arriba, produciéndose un «OUT OF MEMORY» si colisionan ambas.

Otra diferencia es el tamaño de los datos: la pila de máquina almacena datos de 2 bytes y los datos del calculador ocupan 5 bytes.

Los datos alfanuméricos se colocan de la siguiente forma: 1 byte de tipo (0 = matriz, 1 = cadena, otro = literal), dos bytes que señalan la dirección donde se encuentra y otros dos que determinan la longitud de ésta.

Los datos numéricos se pueden almacenar de dos formas: El formato de «pequeño entero» en el que el tercero y cuarto bytes contienen el valor del número y el resto son ceros; y el formato «coma flotante» en el que el primer byte es el exponente, el primer bit del segundo byte el signo y el resto, 31 bits, la mantisa.

Nombre	Hex.	Dec.	
CALCULATE	335BH	13147d	RST 28H



Memoria auxiliar

Las operaciones complejas necesitan manipular muchos datos, para lo que necesitan un lugar de almacenamiento temporal.

La variable del sistema MEMBOT contiene 30 bytes que ofrecen la posibilidad de almacenar hasta 6 datos al mismo tiempo.



La variable MEM es la que indica dónde se sitúa la memoria, de forma que si cambiamos el valor MEM a cualquier lugar diferente de MEMBOT tendremos la posibilidad de multiplicar el espacio de memoria.

La variable BREG se carga inicialmente con el contenido del registro B y es usada como contador en la instrucción **dec-jr-nz.**

Manejo del stack del calculador

Para introducir o sacar datos del calculador existen una serie de rutinas explicadas en las fichas, cuyo número se indica:

Escritura de datos:

EXPT-2-NUM	1C7AH	M-30	DEC-TO-FP	2C9BH	M-40
EXPT-1-NUM	1C82H	M-30	STACK-A	2D28H	M-40
FETCH-NUM	1CDEH	M-30	STACK-BC	2D2BH	M-40
SCANNING	24FBH	M-37	IN-TO-FP	2D3BH	M-41
STK-VAR	2996H	M-38	P-INT-STO	2D8CH	M-41
STK-STORE	2AB6H	M-39	INT-STORE	2D8EH	M-41
STK-DIGIT	2D22H	M-40	STACK-NUM	33B4H	M-43

Lectura de datos:

TWO-PARAM	1E85H	M-32	INT-FETCH	2D7FH	M-41
FIND-INT-1	1E94H	M-32	FP-TO-BC	2DA2H	M-42
FIND-INT-2	1E99H	M-32	FP-DELETE	2DADH	M-42
STK-TO-BC	2307H	M-35	FP-TO-A	2DD5H	M-42
STK-FETCH	2BF1H	M-39	PRINT-FP	2DE3H	M-42

end-calc (Fin de los cálculos) 38H

Este código debe ser siempre el último. Indica el fin de la rutina del calculador.

Entrada: Ninguna.

Salida : Registros: HL = STKEND-5; comienzo del número de lo alto del STK.
DE = STKEND; Sobre el STK.

fp-calc-2 (Cálculo indirecto) 3BH

Efectúa la operación cuyo código se encuentre en BREG (Registro B al llamar a RST 28H).

Ejemplo:

```
LD      B,4     Equivale a:
RST     28H     RST     28H
DEFB    3BH     DEFB    4
```

Argumentos: Según la operación.

Entrada: STK: Según la operación.

Operación	Código		Dirección	
Nombre	**Hex**	**Dec.**	**Hex.**	**Dec.**
end-calc	38H	56d	369BH	13979d
fp-calc-2	3BH	59d	33A2H	13218d
addition	0FH	15d	3014H	12308d
subtract	03H	3d	300FH	12303d
multiply	04H	4d	30CAH	12490d
división	05H	5d	31AFH	12719d
sin	1FH	31d	37B5H	14261d
cos	20H	32d	37AAH	14250d
tan	21H	33d	37BAH	14298d
asn	22H	34d	3833H	14387d
acs	23H	35d	3843H	14403d
atn	24H	36d	37E2H	14306d
get-argt	39H	57d	3783H	14211d

Registros: B = Código de operación.

Salida : Según la operación.

Espacio de trabajo: Según la operación.

MEM usada: Según la operación.

addition (suma) 0FH **subtract** (resta) 03H
multiply (multiplic.) 04H **división** 05H

Efectúa la operación correspondiente con los dos números de lo alto del stack del calculador (STK), que son sustituidos por el resultado. De esta forma el stack resulta reducido.

Argumentos: Ninguno.

Entrada: Alto del STK.: Operando númerico. (sustraendo, divisor).
Dato anterior: Operando numérico. (minuendo, dividendo).

Salida : Alto del STK.: Resultado (número).

sin 1FH **cos** 20H **tan** 21H
asn 22H **acs** 23H **atn** 24H

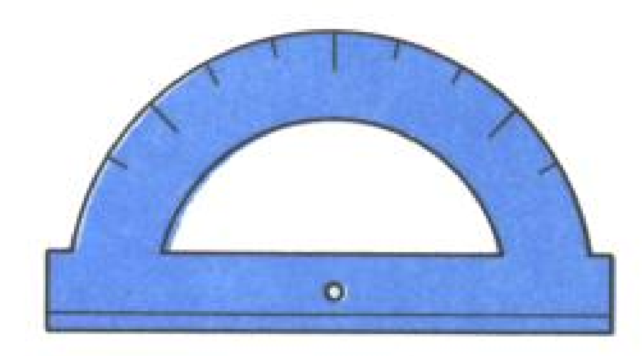
Realiza la función correspondiente sustituyendo el valor inicial por el resultado.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

MEM usada:





get-argt (Obtiene argumento) 39H

Esta rutina obtiene el argumento de SIN X o COS X en un valor que llamaremos V.

En primer lugar calcula Y:

$Y = X/(2 * PI) - INT(X/(2 * PI) + 0.5)$

Posteriormente la rutina retorna con:

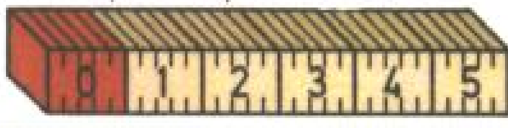
$V = 4 * Y$ si $-1 <= 4 * Y <= 1$
$V = 2 - 4 * Y$ si $1 < 4 * Y < 2$
$V = 4 * Y - 2$ si $-2 <= 4 * Y <= -1$

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: V (argumento).

MEM 0 = 1 si $ABS(4 * Y) > 1$
0 si $ABS(4 * Y) <= 1$

MEM usada:



negate (Complementario: 0—N) 1BH
abs (Valor absoluto) 2AH

Sustituye el valor númerico de lo alto del STK por el resultado de la función correspondiente.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

truncate (Truncamiento) 3AH

Devuelve la parte entera más cercana a 0 de un número cualquiera. Ej.: I(—6.9) = — 6

● Si el entero resultante está entre — 65535 y 65535 lo convierte al formato de «pequeño entero».

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

Int (Parte entera) 27H

Devuelve la parte entera por defecto de un número tanto positivo como negativo. Ej.: INT (— 6.5) = — 7.

Operación	Código		Dirección	
Nombre	**Hex**	**Dec.**	**Hex.**	**Dec.**
negate	1BH	27d	346EH	13422d
abs	2AH	42d	346AH	13418d
truncate	3AH	58d	3214H	12820d
int	27H	39d	36AFH	13999d
to-power	06H	6d	3851H	14417d
sqr	28H	40d	384AH	14410d
exq	26H	38d	36C4H	14020d
ln	25H	37d	3713H	14099d
in	2CH	44d	34A5H	13477d
peek	2BH	43d	34ACH	13484d
usr-no	2DH	45d	34B3H	13491d

● Si el entero resultante está entre — 65535 y 65535 lo convierte al formato de «pequeño entero».

Entrada: Alto del STK.: Operando númerico.

Salida : Alto del STK.: Resultado numérico.
MEM 0 = I (X) si X < 0.

MEM usada: 0 1 2 3 4 5

to-power(potenciación: X↑Y) 06H

Eleva a la potencia que indica el número situado en lo alto del stack del caculador, el número situado anteriormente, siendo sustituidos por el resultado. De esta forma el stack resulta reducido.

Entrada: Alto del STK.: Exponente (número).
Dato anterior: Base (número).

Salida : Alto del STK.: Resultado numérico.

MEM usada:

sqr (raíz cuadrada de número positivo) 28H
exp (antilogaritmo neperiano: e↑X) 26H

Sustituye el valor numérico situado en lo alto del STK por el resultado de la función correspondiente.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

MEM usada:

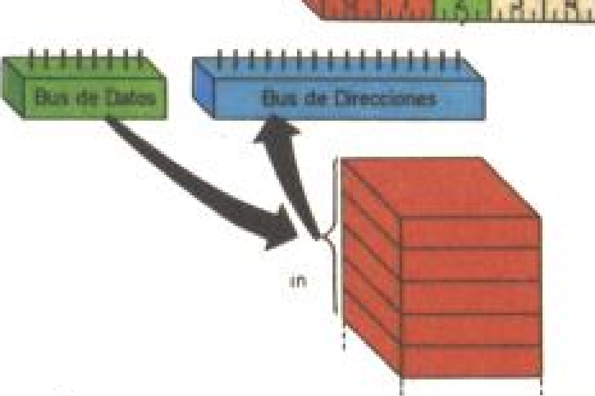
ln (logaritmo neperiano: LN (x) 25H

Sustituye el valor numérico situado en lo alto del STK por su logaritmo neperiano.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

MEM usada:



in 2CH **peek** 2BH **usr-no** (USR numérico) 2DH

Sustituye el número situado en lo alto del STK (redondeado al entero más cercano) por el resultado de la función correspondiente.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

code 1CH **len** 1EH **usr$** 19H

Sustituye el valor alfanumérico situado en lo alto del STK por el resultado numérico de la función correspondiente.

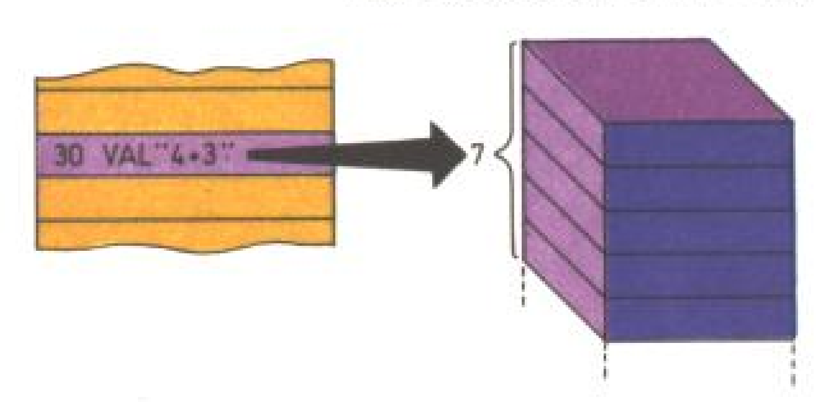
Entrada: Alto del STK.: Operando alfanum.

Salida : Alto del STK.: Resultado numérico.

val 1DH

Sustituye el valor alfanumérico situado en lo alto del STK por su valor numérico.

Entrada: Alto del STK.: Operando alfanum.
Registros: B = 1DH (en caso contrario se efectuaría VAL$).



Operación	Código		Dirección	
Nombre	**Hex**	**Dec.**	**Hex.**	**Dec.**
code	1CH	28d	3669H	13929d
len	1EH	30d	3674H	13940d
usr-$	19H	25d	34BCH	13500d
val	1DH	29d	35DEH	13790d
val$	18H	24d	35DEH	13790d
chr$	2FH	47d	35C9H	13769d
str$	2EH	46d	361FH	13855d
str-add	17H	23d	359CH	13724d

Salida : Alto del STK.: Resultado numérico.

Espacio de trabajo: Cadena alfanumérica más los formatos coma flotante tras los números.

MEM usada: Según el caso.

val$ 18H

Sustituye el valor alfanumérico situado en lo alto del STK por su valor alfanumérico.

Entrada: Alto del STK.: Operando alfanum.
Registros : B < > 1DH (en cuyo caso efectuaría VAL).

Salida : Alto del STK.: Result. alfanumérico.

Espacio de trabajo: Cadena alfanumérica original más los formatos coma flotante tras los números.

MEM usada: Según el caso.

chr$ 2FH

Sustituye el valor numérico situado en lo alto del STK por los parámetros de una cadena alfanumérica de un solo carácter creada en el espacio de trabajo.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Result. alfanumérico.

Espacio de trabajo: Carácter correspondiente.

str$ 2EH

Sustituye el valor numérico situado en lo alto del STK por los parámetros de una cadena alfanumérica creada en el espacio de trabajo.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Result. alfanumérico.

Espacio de trabajo: Cadena alfanumérica.

MEM usada:

str-add (suma de cadenas alfanuméricas) 17H

Sustituye los dos valores alfanuméricos de lo alto del STK por los parámetros de una nueva cadena alfanumérica, compuesta de las dos primeras, creada en el espacio de trabajo.

El stack queda reducido en un dato.

Entrada: Alto del STK.: Operan. alfanumérico.
Dato anterior: Operan. alfanumérico.

Salida : Alto del STK.: Result. alfanumérico.

Espacio de trabajo: Cadena alfanumérica.

or 07H **no-&-no** (número AND número) 08H

X OR Y = X (si Y = 0); ó 1 (si Y < >0)
X AND Y = X (si Y < >0);ó0 (si Y = 0)

El valor de Y es eliminado del STK aunque no borrado (ver «delete» M-49) y el valor de X es mantenido o sustituido por 1 ó 0.

Entrada: Alto del STK.: Operando numér. (Y).
Dato anterior: Operando numér. (X).
Salida : Alto del STK.: Resultado numérico.
Registros : DE = dir. Y = (STKEND).

str-&-no (X$ AND Y) 10H

Si Y < >0 devuelve X$, si Y = 0 devuelve la cadena vacía (longitud 0).

El valor Y es eliminado del STK y X$ se mantiene como estaba o con longitud 0.

Entrada: Alto del STK.: Operando numérico.
Datos anterior: Operando alfanumér.
Salida : Alto del STK.: Resultado alfanumér.
Registros: DE = direc. Y = (STKEND).

Operación		Código		Dirección	
Nombre		**Hex**	**Dec.**	**Hex.**	**Dec.**
or		07H	7d	351BH	13595d
no-&-no		08H	8d	3524H	13604d
str-&-no		10H	19d	352DH	13600d
no-l-eql	< =	09H	9d	353BH	13627d
no-gr-eq	> =	0AH	10d	353BH	13627d
nos-neql	< >	0BH	11d	353BH	13627d
no-grtr	>	0CH	12d	353BH	13627d
no-less	<	0DH	13d	353BH	13627d
nos-eql	=	0EH	14d	353BH	13627d
str-l-eql	< =	11H	17d	353BH	13627d
str-gr-eq	< =	12H	18d	353BH	13627d
strs-neql	< >	13H	19d	353BH	13627d
str-grtr	>	14H	20d	353BH	13627d
str-less	<	15H	21d	353BH	13627d
strs-eql	=	16H	22d	353BH	13627d
greater0	>0	37H	55d	34F9H	13561d
less0	<0	36H	54d	3506H	13574d
not	=0	30H	48d	3501H	13569d
sgn		29H	41d	3492H	13458d

no-l-eql 09H **no-gr-eq**0AH **nos-neql** 0BH
no-grtr 0CH **no-less** 0DH **nos-eql** 0EH

Los dos números situados en lo alto del stack del calculador son sustituidos por el valor 1 ó 0 según la expresión resulte cierta o falsa. El STK resulta reducido.

Entrada: Alto del STK.: Operando numér. (Y).
Dato anterior: Operando numér. (X).
Registros : B = Código de la operación.

Salida : Alto del STK.: Resultado núm. (0/1).

str-l-eql 11H **str-gr-eq** 12H **strs-neql** 13H
str-grtr 14H **str-less** 15H **strs-eql** 16H

Los dos descriptores alfanuméricos situados en lo alto del stack del calculador son sustituidos por el valor 1 ó 0 según la expresión resulte cierta o falsa.
El STK resulta reducido.

Entrada: Alto del STK.: Op. alfanum. (Y$).
Dato anterior: Op. alfanum. (X$).
Registros : B = Código de la operación.

Salida : Alto del STK.: Resultado num.: (0/1).

greater 0 37H **less 0** 36H **not** 30H

El número situado en lo alto del STK es sustituido por 1 ó 0 según resulte cierta o falsa la expresión.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado num. (0/1)

sgn (signo) 29H

El número situado en lo alto del STK es sustituido por —1 si es negativo, por 0 si es 0 ó por 1 si es positivo.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico (-1/0/1).

read-in (lectura de entrada) 1AH

El dato situado en lo alto del STK es considerado como el número de un canal por el que es leído un carácter. Los parámetros de este carácter o de la cadena vacía son colocados en lo alto del stack en sustitución del dato inicial.

Es la rutina utilizada por la función INKEY$. En condiciones normales los canales 0 y 1 nos servirán para leer el teclado.

Entrada: Alto del STK.: Número de canal.

Salida : Alto del STK.: Parámetros alfanum.

Espacio de trabajo: Carácter (si fue recibido).

exchange (intercambio) 01H

Los dos datos situados en lo alto del STK son intercambiados.

Entrada: Alto del STK.: Operando Y.
Dato anterior: Operando X.

Salida : Alto del STK.: Operando X.
Dato anterior: Operando Y.

Operación	Código		Dirección	
Nombre	Hex	Dec.	Hex.	Dec.
read-in	1AH	26d	3645H	13893d
exchange	01H	1d	343CH	13372d
delete	02H	2d	33A1H	13217d
duplicate	31H	49d	33C0H	13248d
n-mod-m	32H	50d	36A0H	13984d
re-stack	3DH	61d	3297H	12951d
e-to-fp	3CH	60d	2D4FH	11599d

delete (suprimir) 02H

El dato situado en lo alto del STK es eliminado de la pila. Este, no obstante, no se borra realmente mientras no se sitúe otro en su lugar, por lo que después de esta función puede ser leído a partir de la dirección señalada por el par de registros DE.

Entrada: Alto del STK.: Cualquier dato.

Salida : Alto del STK.: Dato eliminado.
Registros : DE = Señalando a éste.

dup (duplicación) 31H

Sobre el STK del calculador es colocado un nuevo dato exactamente igual al que en ese momento se encuentre arriba.

Entrada: Alto del STK.: Cualquier dato X.

Salida : Alto del STK.: Dato X.
Dato anterior: Dato X.

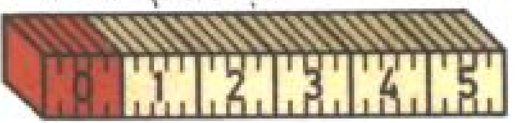
n-mod-n 32H

Dados dos números N y M en lo alto del STK del calculador, éstos son sustituidos por el cociente entero y el resto de N/M.

Entrada: Alto del STK.: Operando numérico M.
Dato anterior: Operando numérico N.

Salida : Alto del STK.: INT (N/M).
Dato anterior: N - M ∗ INT (N-M).
MEM 0 = INT (N/M).

MEM usada:



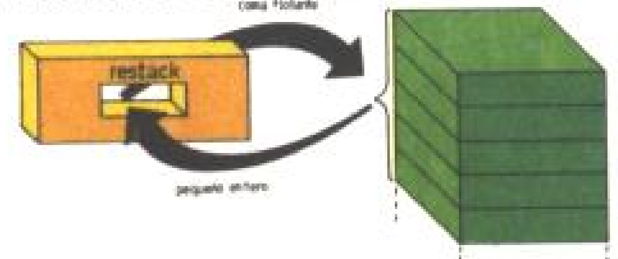
restack (realmacenaje) 3DH

Si el número situado en lo alto del STK se encuentra en el formato de «pequeño entero» es convertido al formato «coma flotante».

Las funciones «int» y «truncate» efectúan la operación inversa.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: n.° en coma flotante.



e-to-fp formato exp. a coma flotante) 3CH

Rutina utilizada por SCANNING para pasar al formato de coma flotante los números en forma exponencial (xEm). «x» debe encontrarse en lo alto del STK y «m» en el acumulador.

Esta rutina debe utilizarse llamando a la dirección 2D4FH (11855d), pues no funciona desde el calculador, debido a que éste modifica A.

jump (salto relativo) 33H

Se produce un salto relativo al código de operación, situado a una distancia indicada por el código siguiente a 33H. Este es considerado como un número en complemento a 2 (— 128 < x < 127).

Argumentos: 1; Distancia de salto.

jump-true (salto si es verdad) 00H

Si el número situado en lo alto del stack del calculador es 1 se produce un salto relativo al código de operación situado a una distancia indicada por el código siguiente a 00H. Este es considerado como un número en complemento a 2 (— 128 < x < 127).

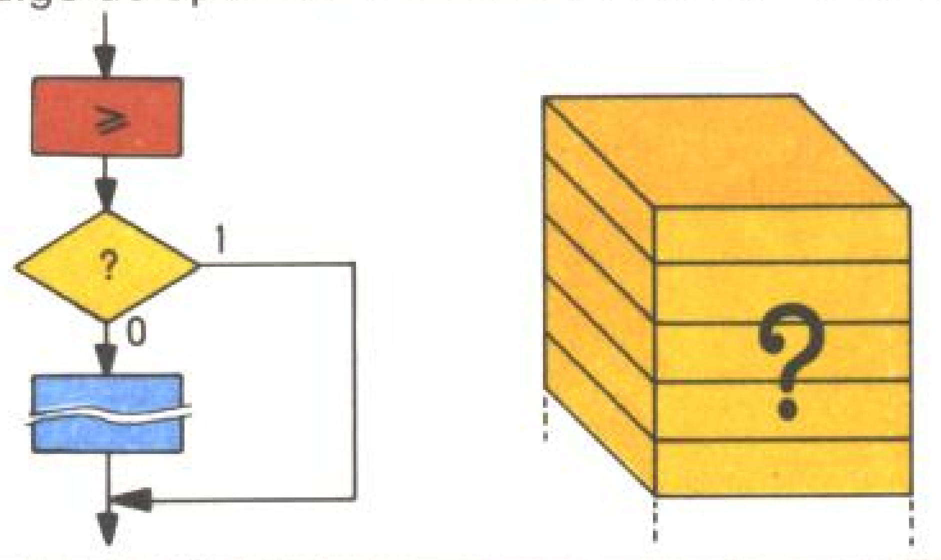
Si en lo alto del STK hubiese un 0 no se produciría este salto.

En ambos casos el número situado en lo alto del STK resulta eliminado.

Argumentos: 1; Distancia de salto.

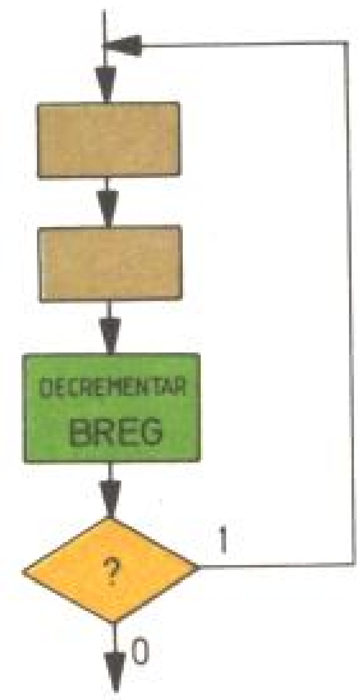
Entrada: Alto del STK.: Número (1/0).

Operación	Código		Dirección	
Nombre	**Hex**	**Dec.**	**Hex.**	**Dec.**
jump	33H	51d	3686H	13958d
jump-true	00H	0d	368FH	13967d
dec-jr-nz	35H	53d	367AH	13946d
stk-zero	A0H	160d	341BH	13339d
stk-one	A1H	161d	341BH	13339d
stk-half	A2H	162d	341BH	13339d
stk-pi/2	A2H	163d	341BH	13339d
stk-ten	A4H	164d	341BH	13339d



dec-jr-nz (dec. y saltar si no es 0) 35H

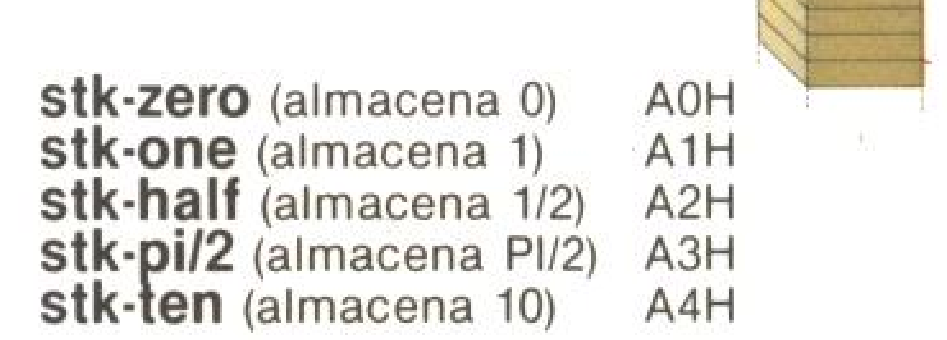
El contenido de la variable BREG es decrementado, si el resultado no es 0 se produce un salto relativo, si resulta 0 no se produce el salto.



Esta rutina es usada por el generador de series (86,88,8C) y, por tanto, también indirectamente por val, sin, cos, tan, asn, acs, atn, ln, exp y sqr.

Puede usarse por el programador teniendo en cuenta que BREG toma el valor del registro B al llamar a RST 28H, pero puede ser modificado por cualquiera de las instrucciones antes citadas.

Argumentos: 1; Distancia de salto.
Entrada: (BREG) como contador.
Salida : (BREG) decrementado.



stk-zero (almacena 0) A0H
stk-one (almacena 1) A1H
stk-half (almacena 1/2) A2H
stk-pi/2 (almacena PI/2) A3H
stk-ten (almacena 10) A4H

El número indicado es almacenado en lo alto de la pila del calculador.
Salida : Alto del STK.: Número almacenado.

stk-data (almacena un dato) 34H

El número indicado por la serie de argumentos que sigue al código de operación es almacenado en la pila del calculador.

El significado de estos argumentos es como sigue: El primer argumento es dividido entre 40H y al cociente se le suma 1 para obtener el número de datos de mantisa. Si el resto de la división no es cero se le suma 50H para obtener el exponente; si el resto fuese 0 el exponente sería el siguiente argumento incrementado también en 50H.

El número final es completado con ceros hasta llegar a los 5 bytes que lo componen.

Ej. = 80H B0H 00H 12H 30H
INT (80H/40H) = 2 ; 2 + 1 = 3 cifras
80Hmod40H = 0 ; ver siguiente dato
B0H + 50H = 0H ; Exponente 0
Mantisa (3 cifras) 00H 12H 30H (+ 1 cero)00H
El número resultante es el «pequeño entero»
3012H = 12306d

Argumentos: Varios.

Operación	Código		Dirección	
Nómbre	**Hex**	**Dec.**	**Hex.**	**Dec.**
stk-data	34H	52d	33C6H	13254d
stk-mem-0	C0H	192d	342DH	13357d
stk-mem-1	C1H	193d	342DH	13357d
stk-mem-2	C2H	194d	342DH	13357d
stk-mem-3	C3H	195d	342DH	13357d
stk-mem-4	C4H	196d	342DH	13357d
stk-mem-5	C5H	197d	342DH	13357d
get-mem-0	E0H	224d	340FH	13327d
get-mem-1	E1H	225d	340FH	13327d
get-mem-2	E2H	226d	340FH	13327d
get-mem-3	E3H	227d	340FH	13327d
get-mem-4	E4H	228d	340FH	13327d
get-mem-5	E5H	229d	340FH	13327d
series-06	86H	134d	3449H	13385d
series-08	88H	136d	3449H	13385d
series-0C	8CH	140d	3449H	13385d

Salida : Alto del STK.: Número almacenado.

stk-mem (cargar en memoria) C0H a C5H

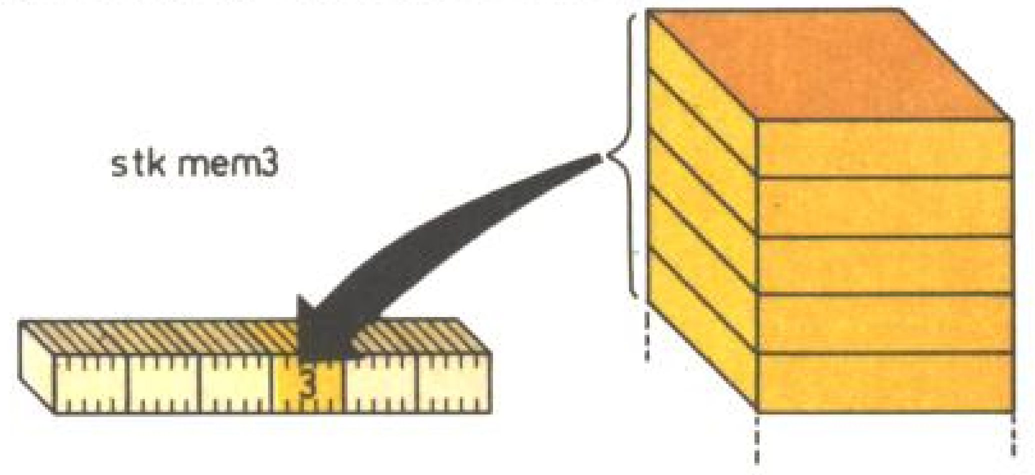
El dato situado en lo alto del STK es copiado en la memoria indicada. Este dato se mantiene también en lo alto del stack del calculador.

La zona de memoria señalada por MEM (generalmente MEMBOT, pero no necesariamente) se compone de 30 bytes que, agrupados de 5 en 5, constituyen las 6 memorias de acceso directo del calculador.

Entrada: Alto del STK.: Dato por guardar.

Salida : Alto del STK.: Permanece el dato.

MEM usada: La determinada por la instrucción.



get-mem (extraer de memoria) E0H a E5H

El dato, situado en la memoria que indique la instrucción, es copiado en lo alto del STK. De esta forma el stack del calculador es ampliado.

Salida : Alto del STK.: Dato extraído.

series-06 86H **series-08** 88H **series-0C** 8CH

Esta rutina genera las series de Chebyshev, que sirven para hallar por aproximación las funciones SIN, ATN, LN y EXP, e indirectamente COS, TAN, ASN, ACS, ↑ y SQR.

Detras del código debe ir el número de datos que exige cada instrucción (6, 8 ó 12), en el mismo formato que el usado en el comando «stk-data».

Argumentos: Múltiples.

Entrada: Alto del STK.: Operando numérico.

Salida : Alto del STK.: Resultado numérico.

MEM usada:

En la serie de rutinas en lenguaje ensamblador, disponemos de utilidades para ampliar la potencia del Basic y de rutinas para usar desde nuestros programas en código máquina.

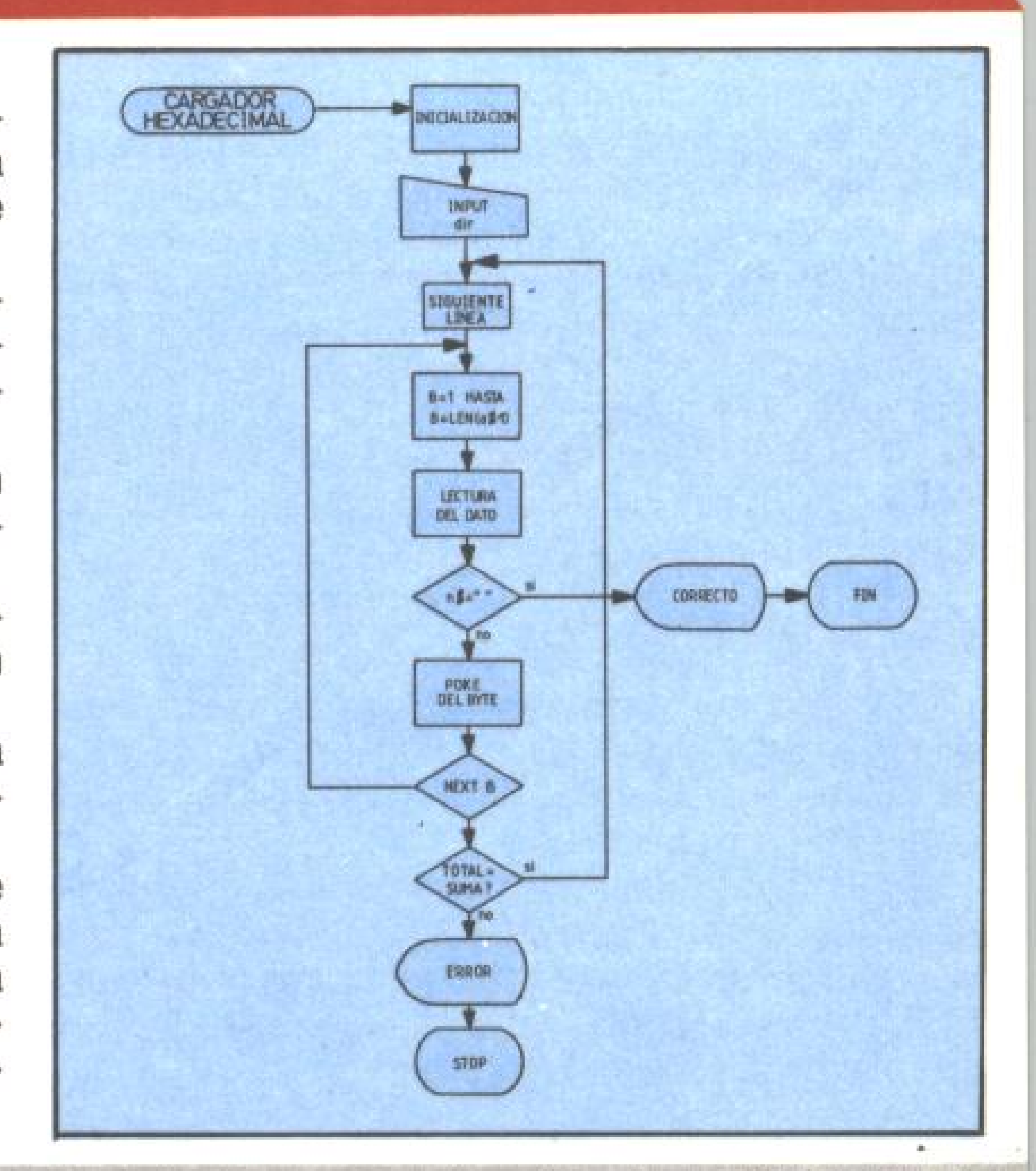
En la descripción de cada rutina se explica cómo se usa y cómo funciona, y se incluye un diagrama de flujo ilustrativo, y el listado en ensamblador con comentarios.

Si la rutina es utilizable por el Basic, incluirá un bloque de instrucciones DATA con el código máquina para cargarlo desde el Basic.

Todas las rutinas están ensambladas en la dirección 60000 mediante la Pseudoinstrucción ORG que se puede variar fácilmente.

Puede tener una primera parte que se encarga de tomar los posibles parámetros proporcionados por el Basic, si es utilizable desde él.

Para acceder desde código máquina a la parte principal de la rutina, que es la que efectúa la operación, puede hacerse una llamada directa mediante la instrucción CALL START, (previamente hay que colocar los parámetros necesarios).



● Para cargar el bloque de DATA con el código máquina, se añade a este programa en basic, el cual realiza el volcado de dicho código en memoria, aceptando la dirección de comienzo, que será 60.000 para las rutinas no reubicables, y la dirección deseada para las rutinas que sí lo son.

Si se produce un error se interrumpe el programa, pudiendo editar directamente la línea en que se ha producido, al habar sido POKEada en la variable de sistema EPPC, dirección 23625, en forma de 2 bytes.

Funcionamiento:1

Se repite un bucle que lee cada línea de DATA en la variable «A$», y la suma de comprobación, en «Total», hasta que el byte hexadecimal sea un espacio, en que termina.

Dentro de este bucle se recorre «A$», realizando el correspondiente POKE en la dirección «dir» del código «byte», y se realiza la suma de comprobación en «suma», que se compara con «Total», para conocer si hay error.

```
1000 REM  CARGADOR HEXADECIMAL
1010 DEF FN N(N$)=CODE N$-48-7*(N$>"9")
1020 CLEAR 59999
1030 LET Linea=0
1040 INPUT "Direccion: ";Dir
1050 LET Linea=Linea+10
1060 RESTORE LINEA
1070 LET Suma=0: READ A$,Total
1080 FOR B=1 TO LEN A$-1 STEP 3
1090 LET N$=A$(B TO B+1)
1100 IF n$(1)=" " THEN  GO TO 1220
1110 LET Byte=16*FN N(N$(1))+FN N(N$(2))
1120 POKE Dir,Byte
1130 LET Dir=Dir+1: LET Suma=Suma+Byte
1140 NEXT B
1150 IF Suma<>Total THEN  GO TO 1170
1160 PRINT "LINEA ";LINEA;" OK.": GO TO 1050
1170 REM ERROR
1180 PRINT  FLASH 1;"Error en linea ";Linea
1190 POKE 23626,INT (Linea/256)
1200 POKE 23625,Linea-256*PEEK 23626
1210 STOP : GO TO 1060
1220 REM CORRECTO
2000 PRINT '"CARGA CORRECTA"
```

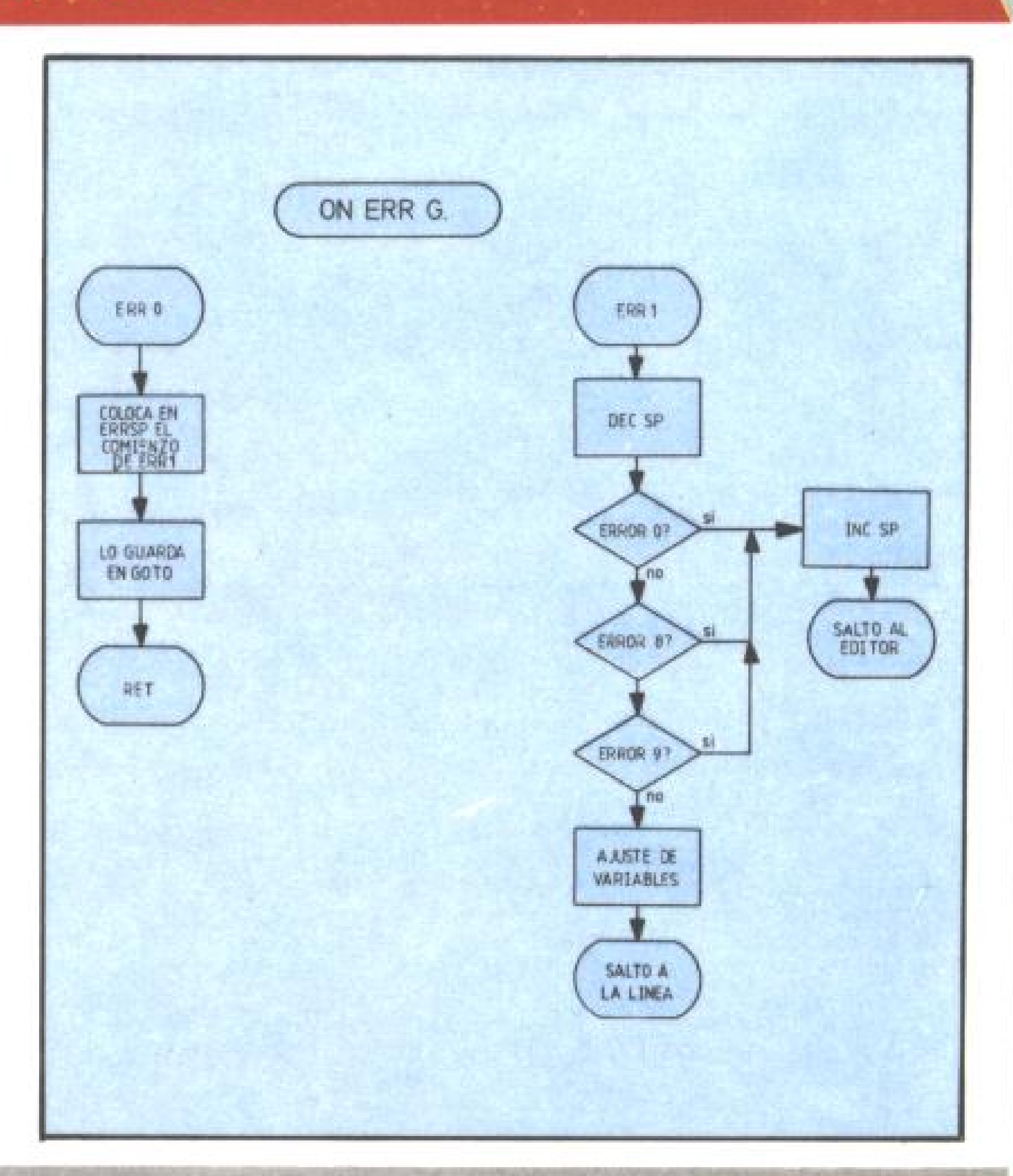
Esta rutina detecta cualquier error excepto «OK», «End of file» y «STOP statement», saltando a la línea Basic deseada, (el número de error, se conoce con la instrucción «PEEK 23681»).

Para ponerla en funcionamiento, una vez cargada en cualquier dirección DIR (es reubicable), debe hacerse al principio del programa, una llamada «RANDOMIZE (línea BASIC en caso de error) + USR DIR.

Funcionamiento:

La primera parte de la rutina, ajusta la variable ERRSP, de tal manera que al ocurrir un error no salte al editor de Basic, sino a la segunda parte de la rutina, y por otro lado toma el número de línea del Basic del Stack del calculador (CALL FINT 2), y lo guarda en la dirección 23738 (GOTOL).

La segunda parte coloca el número de línea en la variable NEWPPC, un 0 en NSPPC y el número de error en ERRNR2 saltando al Basic (CALL STMTR1), excepto si son los errores mencionados arriba, en cuyo caso salta al editor (CALL MAIN4).



```
10 ; *  ON  ERROR  GOTO  *
20 ;
30 ;
40          ORG    60000     ;RUTINA REUBICABLE
50 ;
60 ERR0     LD     HL,ERR1-ERR0;Long. de la rut.
70          ADD    HL,BC     ;Calcula dir. ERR 1
80          EX     DE,HL     ;La transfiere a DE
90          LD     HL,(ERRSP);La guarda en ERRSP
100         LD     (HL),E    ;  (Dir. de salto
110         INC    HL        ;  en caso de error)
120         LD     (HL),D    ;
130         CALL   FINT2     ;Lee del STK no. lin.
140         LD     (GOTOL),BC;Lo guarda en 23728
150         RET    ;         ;Vuelve al BASIC
160 ;
170 ;
180 ERR1    DEC    SP        ;Decrementa STACK
190         DEC    SP        ;
200         LD     A,(IY+0)  ;Carga cod. de error
210         INC    A         ;Lo incrementa
220         CP     #00       ;
230         JR     Z,CONT    ;Salta si es 0 OK
240         CP     #08       ;Salta si es 8
250         JR     Z,CONT    ;  END OF FILE
260         CP     #09       ;Salta si es 9
270         JR     Z,CONT    ;  STOP STATEMENT
280         LD     (ERRNR2),A ;Guarda cod. error
290         LD     (IY+0),#FF ;Error 0   OK
300         LD     HL,(GOTOL) ;Numero de linea
310         LD     (NEWPPC),HL;  a saltar
320         XOR    A         ;
330         LD     (IY+10),A;Primera instruccion
340         SET    7,(IY+1)  ;BASIC ejecutandose
350         JP     STMTR1    ;Salta a la linea
360 ;
370 ;
380 CONT    INC    SP        ;Restablece STACK
390         INC    SP        ;
400         JP     MAIN4     ;Continua el programa
410 ;                        ; deteniendose con el
420 ;                        ; codigo de error
430 ;                        ;  correspondiente
440 ;
450 ;
460 ERRSP   EQU    #5C3D     ;Dir. a salt. en err.
470 GOTOL   EQU    #5CB0     ;23728 VAR. no usada
480 NEWPPC  EQU    #5C42     ;No. de linea a salt.
490 FINT2   EQU    #1E99     ;Lee no. del STK num.
500 STMTR1  EQU    #1B7D     ;Salto prox. instr.
510 MAIN4   EQU    #1303     ;Bucle princip. edit.
520 ERRNR2  EQU    23681     ;VAR. no usada
```

```
10 DATA "21 13 00 09 EB 2A 3D 5C",491
20 DATA "73 23 72 CD 99 1E ED 43",956
30 DATA "B0 5C C9 3B 3B FD 7E 00",966
40 DATA "3C FE 00 28 20 FE 08 28",688
50 DATA "1C FE 09 28 18 32 81 5C",626
60 DATA "FD 36 00 FF 2A B0 5C 22",906
70 DATA "42 5C AF FD 77 0A FD CB",1171
80 DATA "01 FE C3 7D 1B 33 33 C3",899
90 DATA "03 13                  ",22
```

DELETE

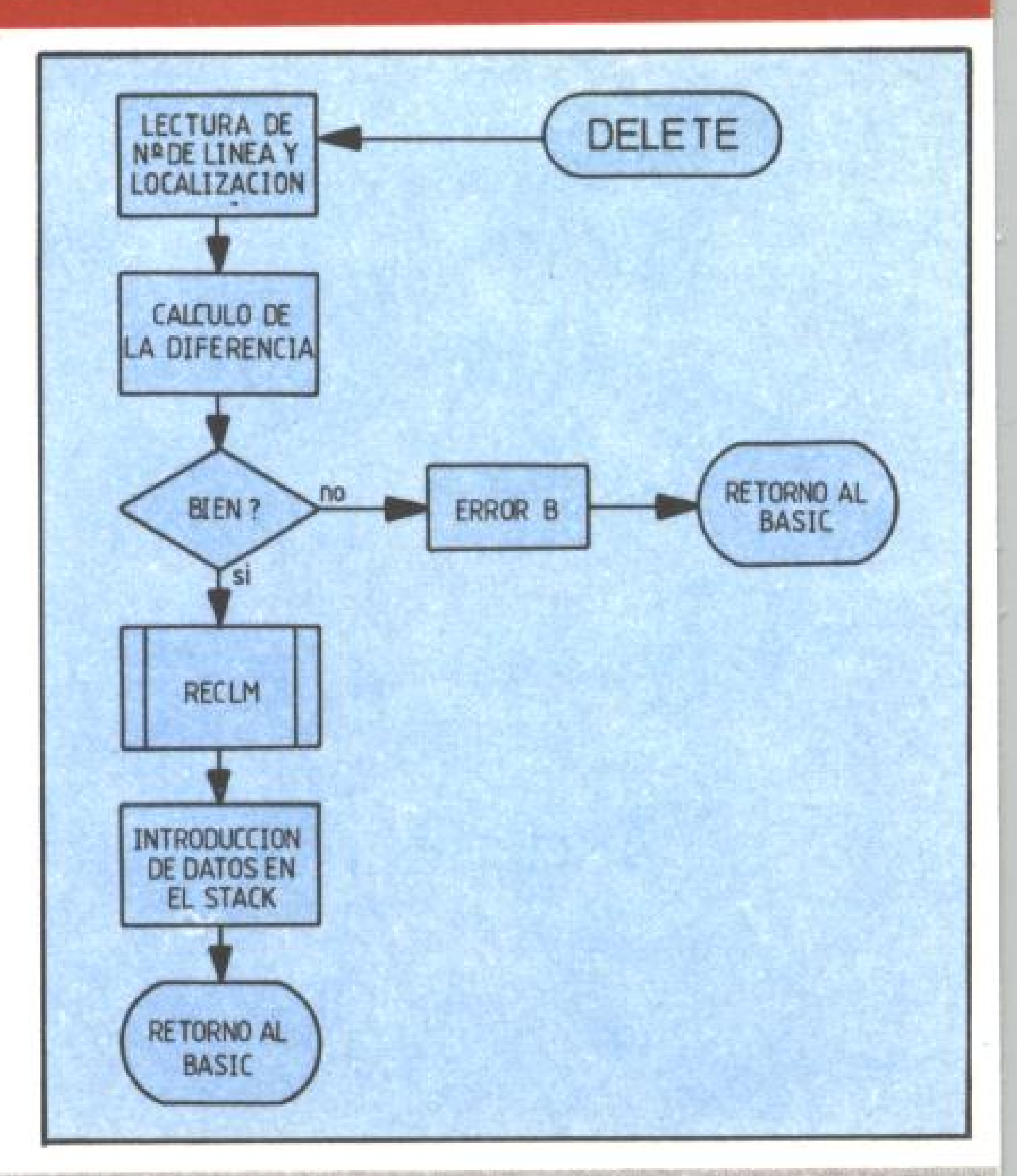
Esta rutina realiza un borrado del basic comprendido entre las líneas N y M, ambas incluidas; para esto, después de cargar la rutina en la dirección DEL que se desee (es reubicable), se hace una llamada de la forma «LET L = N − M * USR DEL».

Al volver al BASIC, la variable L contiene el número de bytes borrados, excepto si M es mayor que N o no existan líneas en ese ámbito, que produce el error «B integer out of range».

Funcionamiento:

Llama dos veces a la subrutina FINT2 asociada con la LINADR, la primera vez con M (última línea a borrar) y la segunda con N (primera línea a borrar); FINT2 recupera los valores M y N del stack y LINADR convierte M y N en dirección de programa para calcular el espacio total entre ambas líneas.

La rutina RECLM1 mueve el bloque posterior del basic (hasta STKEND) para situarlo a continuación del anterior, y ajusta todos los punteros (VARS, etc.) a su nuevo emplazamiento.



```
10 ;  *   DELETE   *
20 ;
30         ORG     60000      ;RUTINA REUBICABLE
40 ;
50         CALL    FINT2      ;Lee M del STK
60         LD      H,B        ;Lo transfiere a HL
70         LD      L,C        ;
80         INC     HL         ;Incrementa no. linea
90         CALL    LINADD     ;Conv. en direccion
100        PUSH    HL         ;Guarda direccion M+1
110        CALL    FINT2      ;Lee N del STK
120        LD      H,B        ;Lo transfiere a HL
130        LD      L,C        ;
140        CALL    LINADD     ;Conv. en direccion
150        POP     DE         ;Recupera dir. M+1
160        EX      DE,HL      ;Intercambia M con N
170        OR      A          ;Carry a 0
180        SBC     HL,DE      ;Longitud a borrar
190        JR      C,ERROR    ;Error si es negativa
200        ADD     HL,DE      ;Restablece HL (M)
210        PUSH    DE         ;Guarda dir. (N)
220        PUSH    HL         ;Guarda dir. (M+1)
230 ;                         ;En DE el primer
240 ;                         ;  byte a borrar
250 ;                         ;En HL siguiente
260 ;                         ;  byte al ultimo
270 ;                         ;  a borrar
280        CALL    RECLM1     ;Borra bloque
290        POP     BC         ;Recupera dir. (M+1)
300        CALL    STKBC      ;La guarda en el STK
310        POP     BC         ;Recupera dir. (N)
320        CALL    STKBC      ;La guarda en el STK
330        LD      BC,1       ;Carga 1 en BC para
340        RET     ;          ;que al ret. al BASIC
350 ;                         ;(RAND n-m*USR 60000)
360 ;                         ;devuelva el num. de
370 ;                         ;bytes borrados
380 ;
390 ERROR  RST     8          ;Error B
400        DEFB    #A         ;Integer out of range
410 ;
420 ;
430 FINT2  EQU     #1E99      ;Lee no. del STK num.
440 LINADD EQU     #196E      ;Busca dir. de linea
450 RECLM1 EQU     #19E5      ;Mueve bloques
460 STKBC  EQU     #2D2B      ;Guarda numero en el
470 ;                         ;  stack numerico
```

```
10 DATA "CD 99 1E 60 69 23 CD 6E",939
20 DATA "19 E5 CD 99 1E 60 69 CD",1048
30 DATA "6E 19 D1 EB B7 ED 52 38",1137
40 DATA "12 19 D5 E5 CD E5 19 C1",1137
50 DATA "CD 2B 2D C1 CD 2B 2D 01",780
60 DATA "01 00 C9 CF 0A          ",419
```

Menaje de pantalla

La forma de llamada es RANDOMIZE USR n+d, siendo n la dirección donde se ubicará la rutina y d el desplazamiento de la subrutina que queremos utilizar para operar con los ficheros de imagen de 2 pantallas, la del sistema y la de trabajo, situada a partir de la dirección 32000.

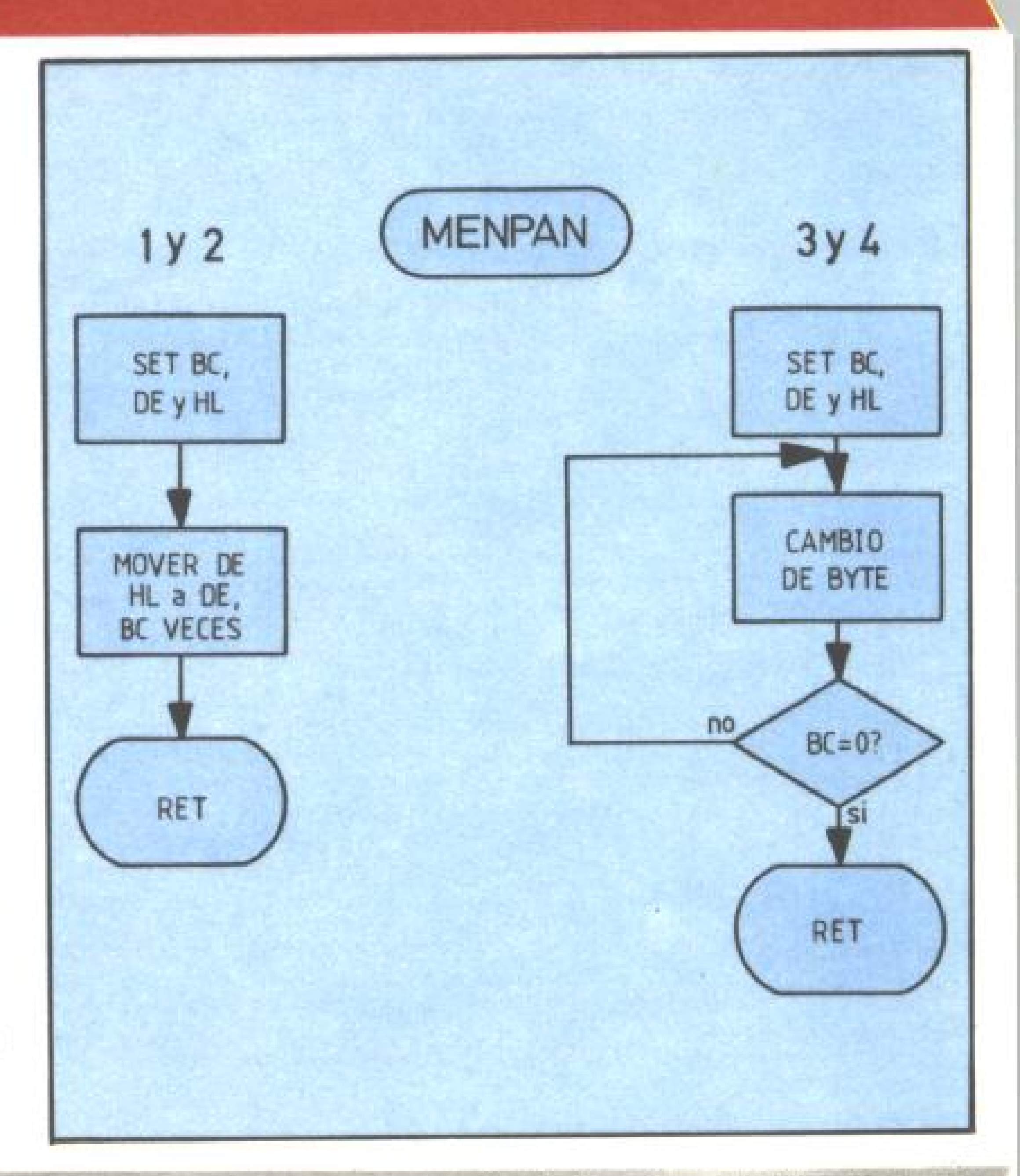
El valor d puede ser 0 (almacenamiento en la pantalla de trabajo), 12 (recuperación de la pantalla de trabajo), 24 (intercambio de ambas pantallas), 47 (mezcla de ambas pantallas).

Para d=47 se puede fijar el modo de mezclado usando la instrucción POKE n+57, códigos 174 (OVER 1 "XOR (HL)"), 182 (OVER 0 "OR (HL)"), 166 (intersección "AND (HL)"), 126 (intercambia el archivo de imagen "LD A,(HL)"), o 47 (INVERSE 1).

Funcionamiento:

Para d=0 y d=12 se carga la dirección inicial de una pantalla en el par HL y la longitud en el par BC, y se transfiere a una zona de memoria cuyo comienzo está especificado por el par DE.

Para d=24 y d=47 se repite un bucle que barre los ficheros de imagen de ambas pantallas, intercambiándolos o mezclándolos.



```
10 ;*  MENAJE DE PANTALLAS *
20 ;
30 ;
40        ORG    60000    ;RUTINA REUBICABLE
50 ;
60 ; ALMACENAMIENTO DE PANTALLA
70 ;
80 START1 LD     HL,16384 ;Com. de la pantalla
90        LD     DE,32000 ;Dir. de la pant. 2
100       LD     BC,6912  ;Longit. de la pant.
110       LDIR   ;        ;Almacena la pantalla
120       RET
130 ;
140 ; RECUPERACION DE PANTALLA
150 START2 LD    HL,32000 ;Dir. de la pant. 2
160       LD     DE,16384 ;Comienzo de la pant.
170       LD     BC,6912  ;Longit. de la pant.
180       LDIR   ;        ;Recupera la pantalla
190       RET
200 ;
210 ; INTERCAMBIO DE PANTALLAS
220 START3 LD    HL,32000 ;Dir. de la pant. 2
230       LD     DE,16384 ;Comienzo de la pant.
240       LD     BC,6912  ;Long. de la pantalla
250 BUCLE1 LD    A,(DE)   ;Intercambia el
260       EX     AF,AF'   ;contenido de
270       LD     A,(HL)   ;la pantalla con la
280       LD     (DE),A   ; pantalla almacenada
290       EX     AF,AF'
300       LD     (HL),A
310       INC    DE       ;Pantalla 1
320       INC    HL       ;Pantalla 2
330       DEC    BC       ;Longitud de pantalla
340       LD     A,B      ;
350       OR     C        ;Comprueba si BC=0
360       JR     NZ,BUCLE1;si no, repite BUCLE1
370       RET
380 ;
390 ;  MEZCLA DE PANTALLAS
400 START4 LD    HL,32000 ;Dir. de la pant. 2
410       LD     DE,16384 ;Com. del DISP.FILE
420       LD     BC,6144  ;Long. DISP.FILE
430 BUCLE2 LD    A,(DE)   ;Cont. del DISP.FILE
440 MODO  XOR    (HL)     ;XOR con la pant. 2
450       LD     (DE),A   ;Result. al DISP.FILE
460       INC    DE       ;DISPLAY FILE
470       INC    HL       ;Segunda pantalla
480       DEC    BC       ;Long. del DISP.FILE
490       LD     A,B      ;
500       OR     C        ;Comprueba si BC=0
510       JR     NZ,BUCLE2;si no, repite BUCLE2
520       RET
```

```
10 DATA "21 00 40 11 00 7D 01 00",240
20 DATA "1B ED B0 C9 21 00 7D 11",816
30 DATA "00 40 01 00 1B ED B0 C9",706
40 DATA "21 00 7D 11 00 40 01 00",240
50 DATA "1B 1A 08 7E 12 08 77 13",351
60 DATA "23 0B 78 B1 20 F3 C9 21",852
70 DATA "00 7D 11 00 40 01 00 18",231
80 DATA "1A AE 12 13 23 0B 78 B1",580
90 DATA "20 F6 C9                ",479
```

Giro Vertical

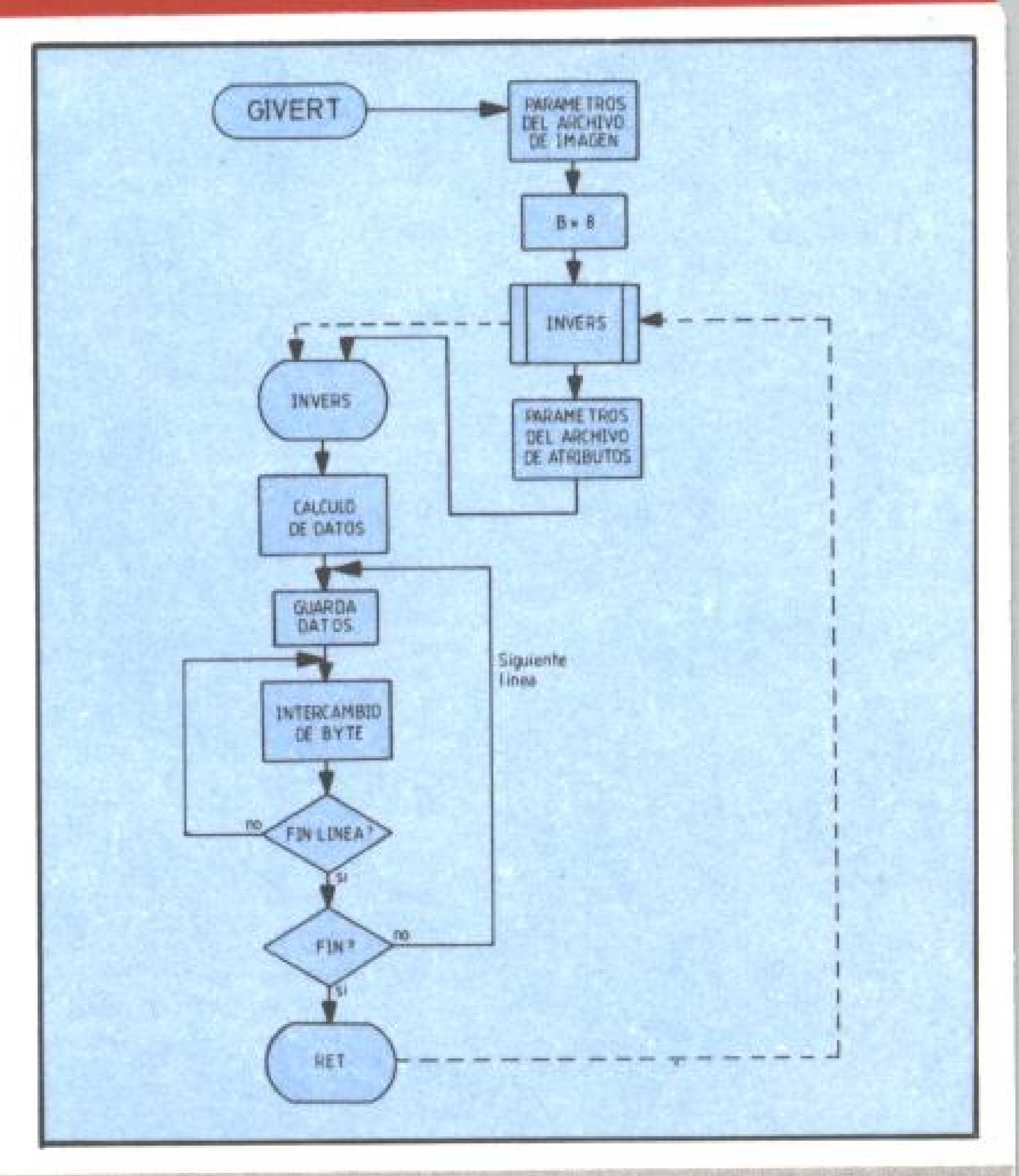
Sabiendo que la pantalla del Spectrum ocupa 6912 bytes (incluyendo atributos), y que está dividida en tres partes de 2304 bytes cada una, se podrá realizar un giro horizontal de 1/3, 2/3 o la pantalla completa en sentido longitudinal (el primer tercio es el superior). La forma de llamada es la usual: RANDOMIZE USR n, siendo n la dirección a partir de la cual se situará la rutina.

Podemos elegir la inversión de 1/3, 2/3 o la pantalla completa utilizando la instrucción Basic POKE n+1,h pudiendo tener h los valores 1, 2 o 3 según las opciones respectivas antes indicadas.

Funcionamiento:

En la línea 60 (LD B, 3) es donde se situará el número resultante de la instrucción POKE realizada anteriormente.

A continuación se intercambian una de las 8 líneas de «pixels» de cada carácter por las del correspondiente opuesto (CALL INVERS), y se realiza el correspondiente ajuste en el fichero de atributos (INVERS).



```
 10 ; *  GIRO VERTICAL  *
 20 ;
 30 ;B: tercios:1,2,3
 40 ;
 50         ORG    60000   ;RUTINA NO REUBICABLE
 60         LD     B,3     ;Pantalla completa
 70 START   LD     HL,16384 ;Comi. de la pantalla
 80         LD     C,32    ;Ancho de linea
 90         PUSH   BC      ;  lo guarda
100         PUSH   HL      ;Guarda com. pantalla
110         SLA.   B       ;B=No. de lineas
120         SLA    B
130         SLA    B
140         CALL   INVERS  ;Invierte fichero
150         POP    HL      ;Recup. com. de pant.
160         LD     DE,#1800 ;Longitud del DISP.FILE
170         ADD    HL,DE   ;Comien. fich. atrib.
180         POP    BC      ;Rec.no. ter. y ancho
190 INVERS  LD     D,B
200         LD     E,0     ;DE=long. a invertir
210         PUSH   BC      ;Gua.no. ter. y ancho
220         PUSH   HL      ;Guarda com. fichero
230         ADD    HL,DE   ;Ultimo byte
240         LD     B,0
250         XOR    A       ;Carry a 0
260         SBC    HL,BC   ;Resta ancho
270         EX     DE,HL   ;DE=final-32
280         POP    HL      ;Comienzo del fichero
290         POP    BC      ;Lineas,ancho
300         SLA    B       ;
310         SLA    B       ;B*4=altura/2
320 BUCLE1  PUSH   BC      ;  lo guarda
330 BUCLE2  LD     A,(HL)
340         PUSH   AF
350         LD     A,(DE)  ;Cambia el contenido
360         LD     (HL),A  ;de DE por
370         POP    AF
380         LD     (DE),A  ;el contenido de HL
390         INC    HL
400         INC    DE
410         DEC    C       ;Ancho
420         JR     NZ,BUCLE2
430         POP    BC
440         PUSH   BC
450         LD     B,0
460         SLA    C       ;C=long. de 2 lineas
470         EX     DE,HL
480         SBC    HL,BC   ;Dec. DE en 2 lineas
490         EX     DE,HL
500         POP    BC      ;Rec. mitad de altura
510         DJNZ   BUCLE1
520         RET
```

```
10 DATA "06 03 21 00 40 0E 20 C5",349
20 DATA "E5 CB 20 CB 20 CB 20 CD",1139
30 DATA "78 EA E1 11 00 18 19 C1",838
40 DATA "50 1E 00 C5 E5 19 06 00",567
50 DATA "AF ED 42 EB E1 C1 CB 20",1366
60 DATA "CB 20 C5 7E F5 1A 77 F1",1189
70 DATA "12 23 13 0D 20 F5 C1 C5",752
80 DATA "06 00 CB 21 EB ED 42 EB",1015
90 DATA "C1 10 E7 C9             ",641
```

Giro Horizontal

Esta rutina realiza un giro de la pantalla tomando como eje una línea vertical situada en el centro de la misma.

La forma de llamada es la usual, es decir:

RANDOMIZE USR n

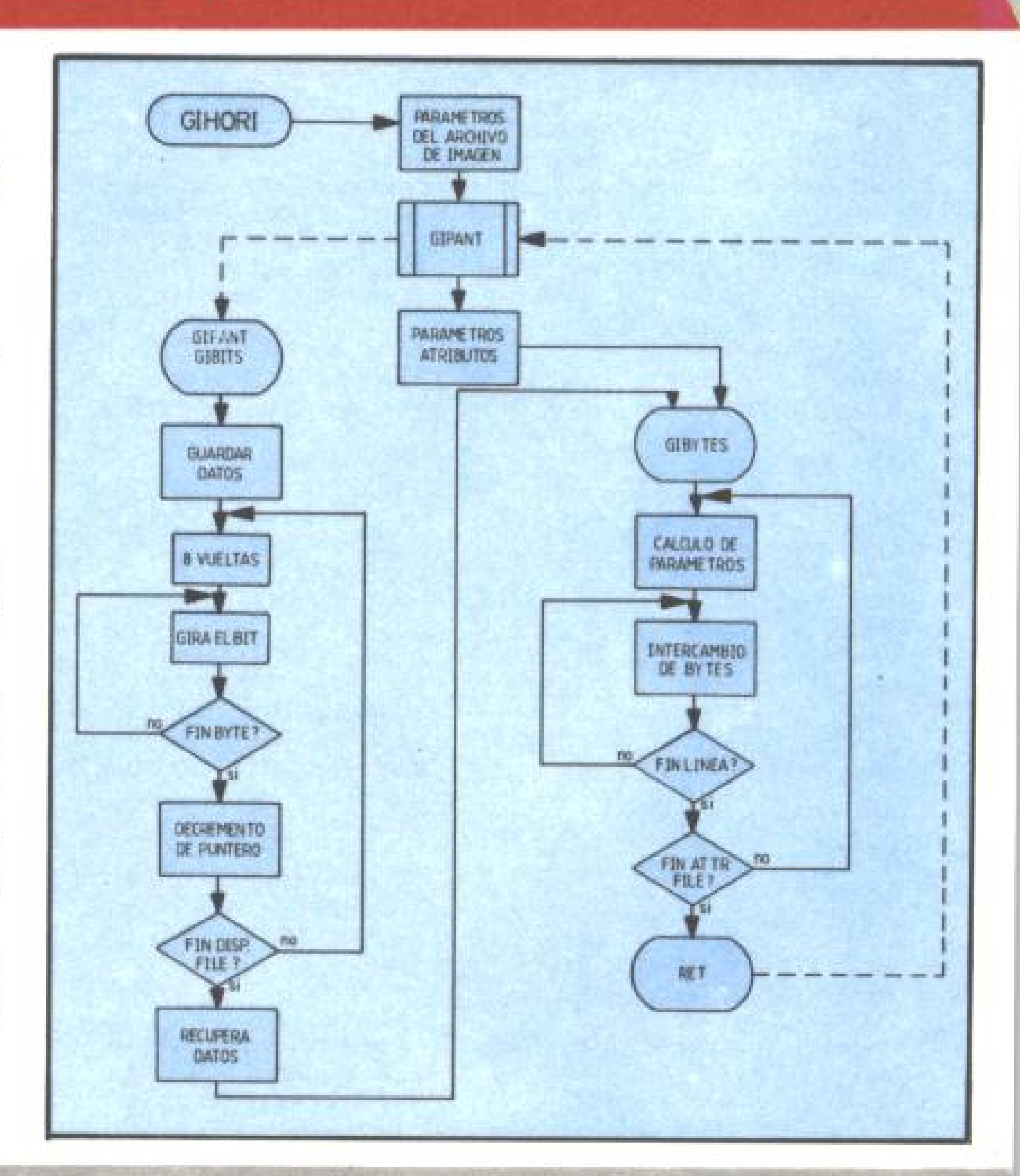
siendo n la dirección a partir de la cual se ha situado la rutina (es relocatable).

Funcionamiento:

Utiliza la subrutina llamada GIPANT compuesta a su vez por otras dos subrutinas cuyos nombres son GIBITS y GBYTES.

La primera parte de la rutina trabaja en el fichero de pantalla, invirtiendo cada una de las 8 líneas de puntos de cada carácter sobre sí mismas, bit a bit (GIBITS), trasladándolas después a su dirección definitiva, al otro lado de la pantalla (GBYTES).

Por último intercambiará los atributos de los caracteres (CALL GBYTES), localizando su dirección en el fichero de atributos.



```
10 ; *  GIRO HORIZONTAL  *
20        ORG    60000   ;RUTINA NO REUBICABLE
30 START  LD     HL,16384 ;Comienzo de pantalla
40        LD     DE,#1800 ;Long. DISPLAY FILE
50        CALL   GIPANT  ;Gira DISPLAY FILE
60        LD     DE,#300 ;Long. archivo atrib.
70        JR     GBYTES  ;Gira arch. atributos
80 GIPANT
90 GIBITS PUSH   HL      ;Com. de pantalla
100       PUSH   DE      ;Long. arch. atribut.
110 BUCLE1 LD    B,8     ;No. de bits por byte
120       LD     A,(HL)
130 BUCLE2 RLA   ;       ;Extrae bit
140       RR     (HL)    ;Guarda bit
150       DJNZ   BUCLE2
160       INC    HL      ;Puntero
170       DEC    DE      ;Longitud
180       LD     A,E
190       OR     D
200       JR     NZ,BUCLE1;6144 vueltas
210       POP    DE      ;Recupera longitud
220       POP    HL      ;Recupera comienzo
230       LD     C,32    ;Anchura de linea
240 GBYTES PUSH  HL      ;Puntero
250       PUSH   DE      ;Longitud
260       PUSH   BC      ;Anchura
270       LD     E,L
280       LD     D,H     ;Transfiere HL a DE
290       ADD    HL,BC   ;Incrementa anchura
300       DEC    HL      ;Puntero A
310       SRL    C       ;C/2
320 BUCLE3 LD    A,(HL)
330       LD     B,A     ;Cambia
340       LD     A,(DE)  ;contenido DE
350       LD     (HL),A  ;por cotenido de HL
360       LD     A,B
370       LD     (DE),A
380       DEC    HL          ;Puntero A
390       INC    DE          ;Puntero B
400       DEC    C           ;Ancho divid. entre 2
410       JR     NZ,BUCLE3
420       POP    BC          ;Ancho
430       POP    HL          ;Puntero
440       OR     A           ;Carry a 0
450       SBC    HL,BC       ;Resta ancho
460       EX     DE,HL       ;Lo transfiere a DE
470       POP    HL          ;Puntero
480       ADD    HL,BC       ;Suma ancho
490       LD     A,D
500       OR     E           ;Continua el bucle
510       JR     NZ,GBYTES;  si DE<>0
520       RET    ;           ;Si DE=0 fin
```

```
10 DATA "21 00 40 11 00 18 CD 6E",453
20 DATA "EA 11 00 03 18 14 E5 D5",740
30 DATA "06 08 7E 17 CB 1E 10 FB",663
40 DATA "23 1B 7B B2 20 F2 D1 E1",1071
50 DATA "0E 20 E5 D5 C5 5D 54 09",871
60 DATA "2B CB 39 7E 47 1A 77 78",765
70 DATA "12 2B 13 0D 20 F5 C1 E1",788
80 DATA "B7 ED 42 EB E1 09 7A B3",1256
90 DATA "20 E0 C9                ",457
```

Decimal a BC

Esta rutina sirve para leer un número decimal escrito en código ASCII y guardar el valor en el par de registros BC.

Puede utilizarse para enviar argumentos numéricos desde el Basic. Este número deberá escribirse en una sentencia REM al comienzo de la siguiente línea en que se encuentre la llamada a código máquina.

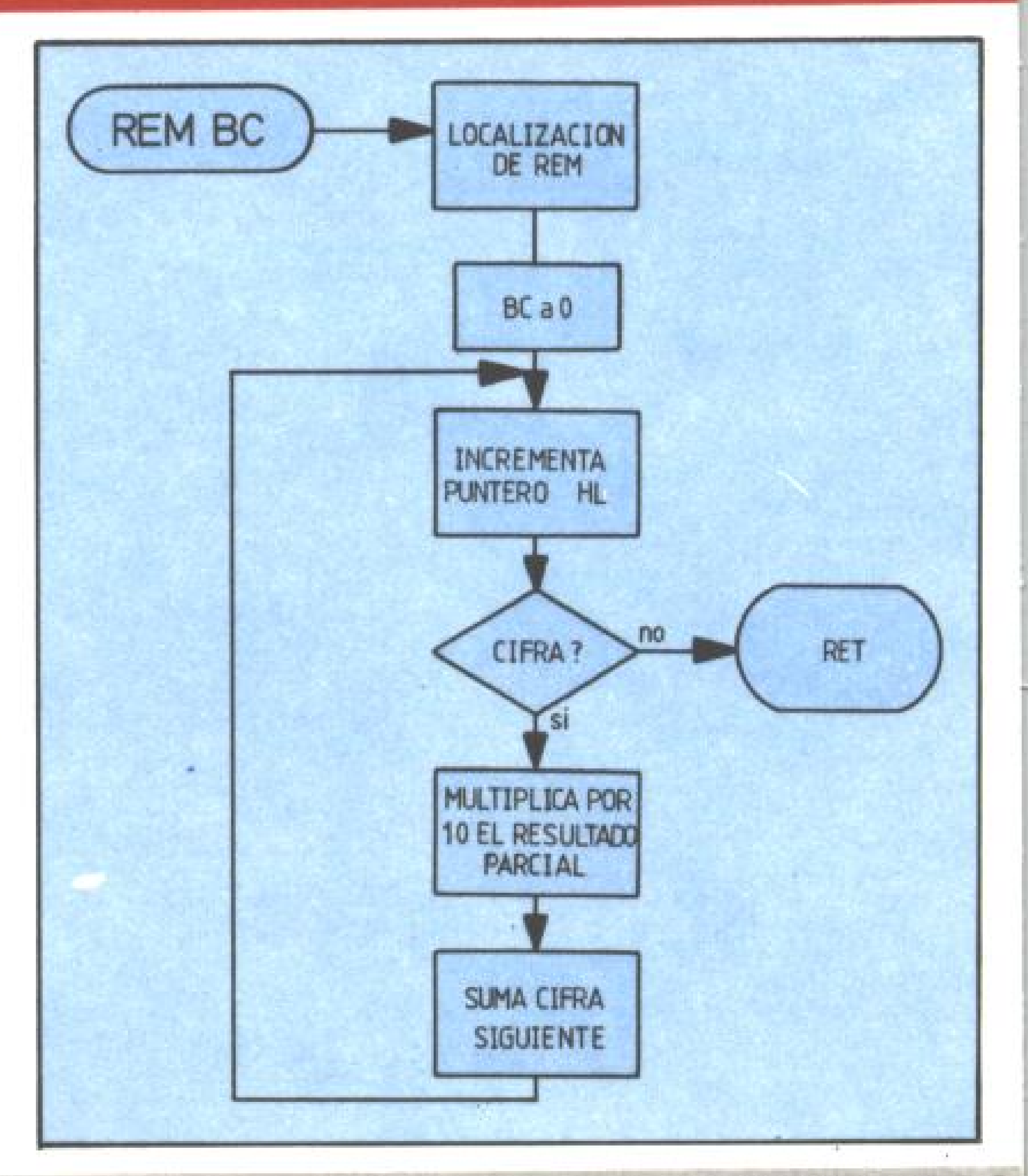
Funcionamiento:

En primer lugar localiza el comienzo de la línea siguiente y lo incrementa en 4 para situarse en la sentencia REM.

A continuación pone BC a cero y lo utiliza de acumulador provisional convirtiendo el número de la siguiente forma:

A cada vuelta multiplica por 10 el resultado parcial acumulado en BC y le suma la cifra siguiente.

La rutina finaliza al encontrar un código que no corresponda a una cifra decimal.



```
10 ;  *  DECIMAL a BC  *
20 ;
30         ORG     60000    ;RUTINA REUBICABLE
40         LD      HL,(NXTLIN);Dir. sig. linea
50         INC     HL       ;Suma 4 a HL para
60         INC     HL       ;  localizar la
70         INC     HL       ;  sentencia REM
80         INC     HL       ;
90 ;                        ;HL ler byte antes de
100 ;                       ;  la primera cifra
110 ;
120 ;
130 START  LD      BC,0     ;Contador a 0
140 BUCLE  INC     HL       ;Proxima cifra
150        LD      A,(HL)   ;La carga en A
160        OR      A        ;Pone a 0 el carry
170        SBC     A,48     ;Conv. ASCII en dec.
180        RET     C        ;
190        CP      10       ;Retorna si no es un
200        RET     NC       ;  numero
210        PUSH    HL       ;Guarda HL
220 ;
230 ;  HL=BC*10
240        LD      H,B      ;Transfiere BC a HL
250        LD      L,C
260        ADD     HL,HL    ;HL*2
270        LD      B,H      ;Transfiere a BC HL*2
280        LD      C,L      ;
290        ADD     HL,HL    ;HL*4
300        ADD     HL,HL    ;HL*8
310        ADD     HL,BC    ;HL*10
320 ;
330 ;  SUMA A HL LA CIFRA SIGUIENTE
340 ;
350        LD      E,A      ;Transfiere A a DE
360        LD      D,0
370        ADD     HL,DE    ;Suma a HL la
380 ;                       ;proxima cifra
390        LD      B,H      ;Transfiere a BC el
400        LD      C,L      ;  valor de HL
410        POP     HL       ;Recupera puntero
420        JR      BUCLE    ;Siguiente cifra
430 ;
440 ;
450 NXTLIN EQU     23637    ;Comienzo de la
460 ;                       ;  proxima linea
```

```
10 DATA "2A 55 5C 23 23 23 23 01",360
20 DATA "00 00 23 7E B7 DE 30 D8",830
30 DATA "FE 0A D0 E5 60 69 29 44",1011
40 DATA "4D 29 29 09 5F 16 00 19",310
50 DATA "44 4D E1 18 E5         ",623
```

Borrado de ventanas

Esta rutina realiza un borrado en la pantalla de «h» cuadrados de alto por «a» de ancho.

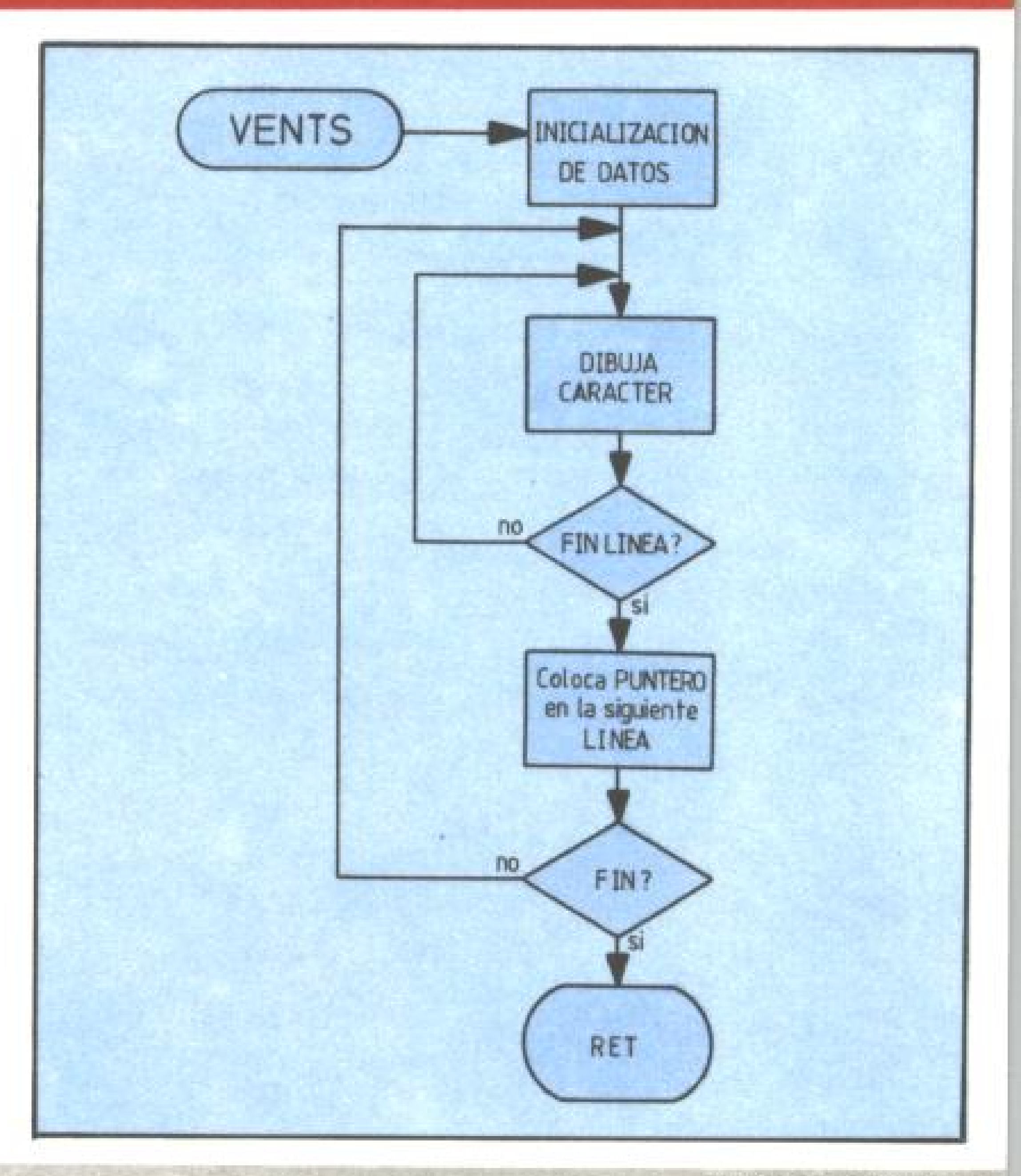
Se sitúa el punto de partida mediante un PRINT AT l,c y tomando esta coordenada como la esquina superior-izquierda| de un rectángulo, se procede a la ejecución de la rutina cuya forma de llamada es: RANDOMIZE h+a * USR n, siendo n la dirección donde se situará la rutina.

Cambiando el color de la tinta, puede ser útil para dibujar rectángulos en la pantalla.

Funcionamiento:

La rutina sitúa en el Acumulador el carácter que va a rellenar el rectángulo (el 32=espacio) y llama a la subrutina de la ROM RST10h.

En BUCLE2 se realiza el borrado de línea, y cuando ésta acaba se sitúa el puntero al principio de la línea siguiente del rectángulo, llamando a RST10h con los valores 22 (AT), 24-H (línea) y 32-L (columna) volviendo a BUCLE1 tantas veces como líneas haya.



```
 10 ; *  BORRADO DE VENTANAS  *
 20        ORG    60000   ;RUTINA REUBICABLE
 30        CALL   FINT1   ;Lee del STK el ancho
 40        PUSH   AF      ;Lo guarda
 50        CALL   FINT1   ;Lee del STK el alto
 60        LD     B,A     ;B=alto
 70        POP    AF
 80        LD     C,A     ;C=ancho
 90        PUSH   BC      ;Guarda dimensiones
100        LD     A,0
110        CALL   STKA    ;Equilibra el
120        LD     A,0     ;  stack numerico
130        CALL   STKA
140        POP    BC      ;Recupera dimensiones
150 ;
160 ;                     ;B=alto  C=ancho
170 START  LD     HL,(SPOSN);Coord. del AT
180 BUCLE1 PUSH   BC      ;Guarda dimensiones
190        LD     A,C     ;Ancho
200        PUSH   HL      ;Guarda coord. del AT
210 ;
220 BUCLE2 PUSH   AF      ;Guarda ancho
230        LD     A,32    ;Cod.ASCII del espac.
240        RST    #10
250        POP    AF      ;Ancho
260        DEC    A
270        JR     NZ,BUCLE2
280 ;
290        LD     A,22    ;Codigo del AT
300        RST    #10
310        POP    HL      ;Coordenadas del AT
320        DEC    H       ;2*y-Linea
330        PUSH   HL      ;Guarda coodenadas
340        LD     A,24
350        SUB    H       ;A=Linea
```

```
360        RST    #10
370        POP    HL      ;Coordenadas
380        PUSH   HL      ;Las guarda
390        LD     A,33
400        SUB    L       ;A=Columna
410        RST    #10
420 ;
430        POP    HL      ;Recupera pos. cursor
440        POP    BC      ;Recupera dimensiones
450        DJNZ   BUCLE1  ;Nueva linea
460        RET
470 ;
480 ;
490 FINT1  EQU    #1E94   ;Lee no. del STK num.
500 STKA   EQU    #2D28   ;Guarda A en STK num.
510 SPOSN  EQU    23688   ;Parametros PRINT
```

```
10 DATA "CD 94 1E F5 CD 94 1E 47",1082
20 DATA "F1 4F C5 3E 00 CD 28 2D",869
30 DATA "3E 00 CD 28 2D 2A 88 5C",622
40 DATA "C1 C5 79 E5 F5 3E 20 D7",1294
50 DATA "F1 3D 20 F8 3E 16 D7 E1",1106
60 DATA "25 E5 3E 18 94 D7 E1 E5",1169
70 DATA "3E 21 95 D7 E1 C1 10 E1",1118
80 DATA "C9                     ",201
```

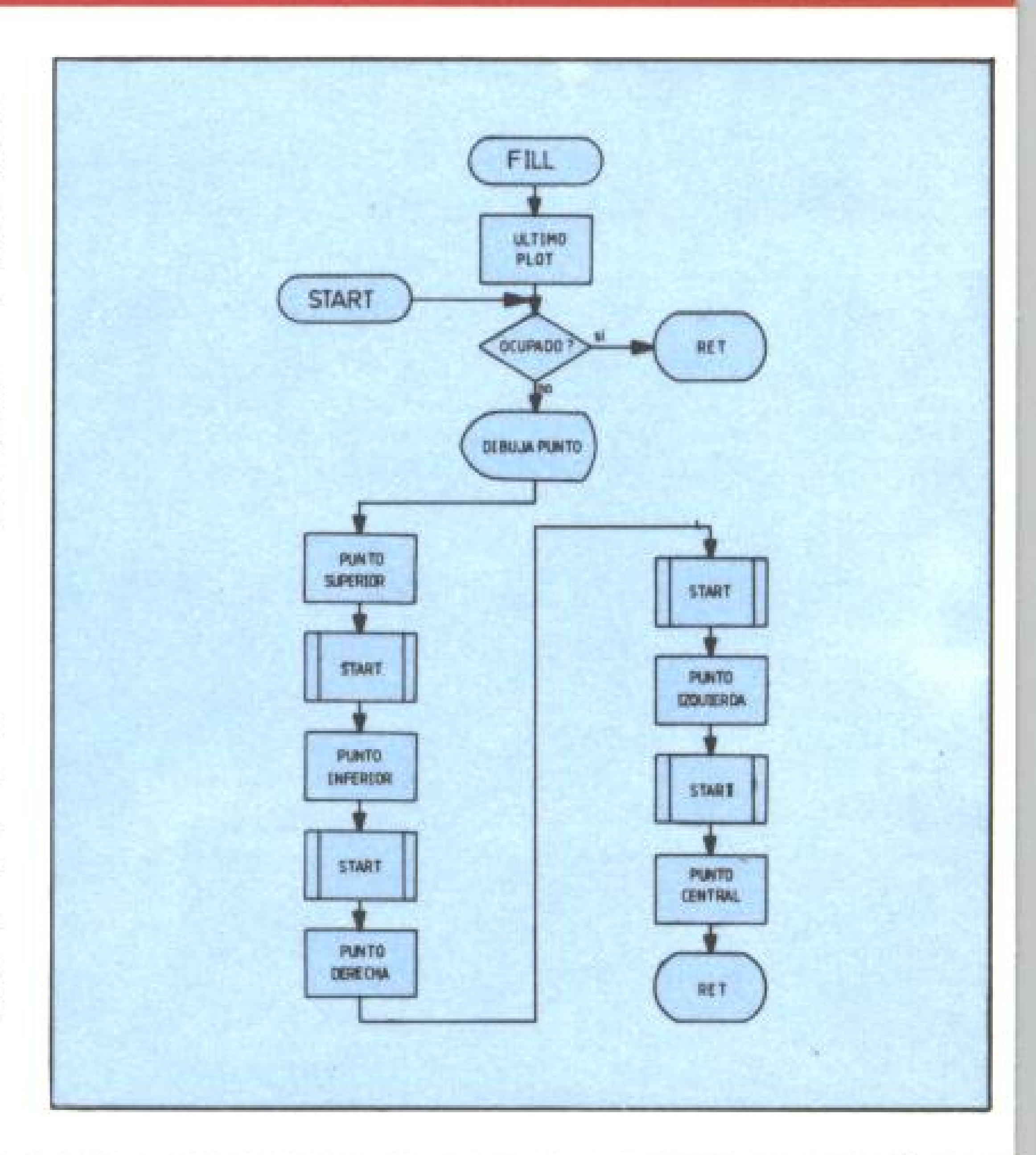
FILL (Rellenado de figuras)

Por medio de esta rutina podremos rellenar cualquier figura por complicada que sea. Para ello, deberemos hacer PLOT INVERSE 1; X, Y: RANDOMIZE USR 60000, donde X e Y son las coordenadas de cualquier punto interior a la figura.

Debido al extremo cuidado que pone para no dejar ningún punto en blanco ocupa mucho stack. Por ello aunque funciona bien en figuras muy complicadas, puede producir un «OUT OF MEMORY» en figuras grandes.

Funcionamiento:

La rutina guarda en BC las coordenadas del último punto trazado, hace una llamada a la rutina POINT, de la ROM, y lee en el stack numérico el resultado, retornando si el punto está ocupado. En caso contrario entra en un bucle autorepetido, en el que la rutina se llama a sí misma para rellenar los cuatro puntos de alrededor de cada punto, y así, sucesivamente.



```
 10 ; ** RELLENADO DE FIGURAS ** (FILL)
 20 ;
 30 ;
 40          ORG    60000     ;RUTINA NO REUBICABLE
 50 ;
 60          LD     BC,(COORDS);Ultimo PLOT
 70 ;
 80 ;
 90 START    PUSH   BC        ;Guarda coordenadas
100          CALL   POINT     ;POINT (C,B)
110          CALL   FINT1     ;Lee POINT del STK
120          POP    BC        ;Recupera coordenadas
130          CP     0         ;Retorna si el punto
140          RET    NZ        ;  esta dibujado
150 ;
160 ;
170 ;
180          PUSH   BC        ;Guarda coordenadas
190          CALL   PLOT      ;PLOT C,B
200          POP    BC        ;Recupera coordenadas
210 ;
220          INC    B         ;Punto superior
230          CALL   START
240 ;
250          DEC    B
260          DEC    B         ;Punto inferior
270          CALL   START
280 ;
290          INC    B
300          INC    C         ;Punto derecha
310          CALL   START
320 ;
330          DEC    C
340          DEC    C         ;Punto izquierda
350          CALL   START
360 ;
370          INC    C         ;Punto central
380          RET
390 ;
400 ;
410 ;
420 COORDS EQU   23677     ;Coordenadas del PLOT
430 POINT  EQU   #22CE     ;Gda. en STK POINT
440 FINT1  EQU   #1E94     ;Lee en A el STK num.
450 PLOT   EQU   #22E5     ;Dibuja un punto
```

```
10 DATA "ED 4B 7D 5C C5 CD CE 22",1171
20 DATA "CD 94 1E C1 FE 00 C0 C5",1219
30 DATA "CD E5 22 C1 04 CD 64 EA",1204
40 DATA "05 05 CD 64 EA 04 0C CD",770
50 DATA "64 EA 0D 0D CD 64 EA 0C",911
60 DATA "C9                     ",201
```

Bold y Double strike

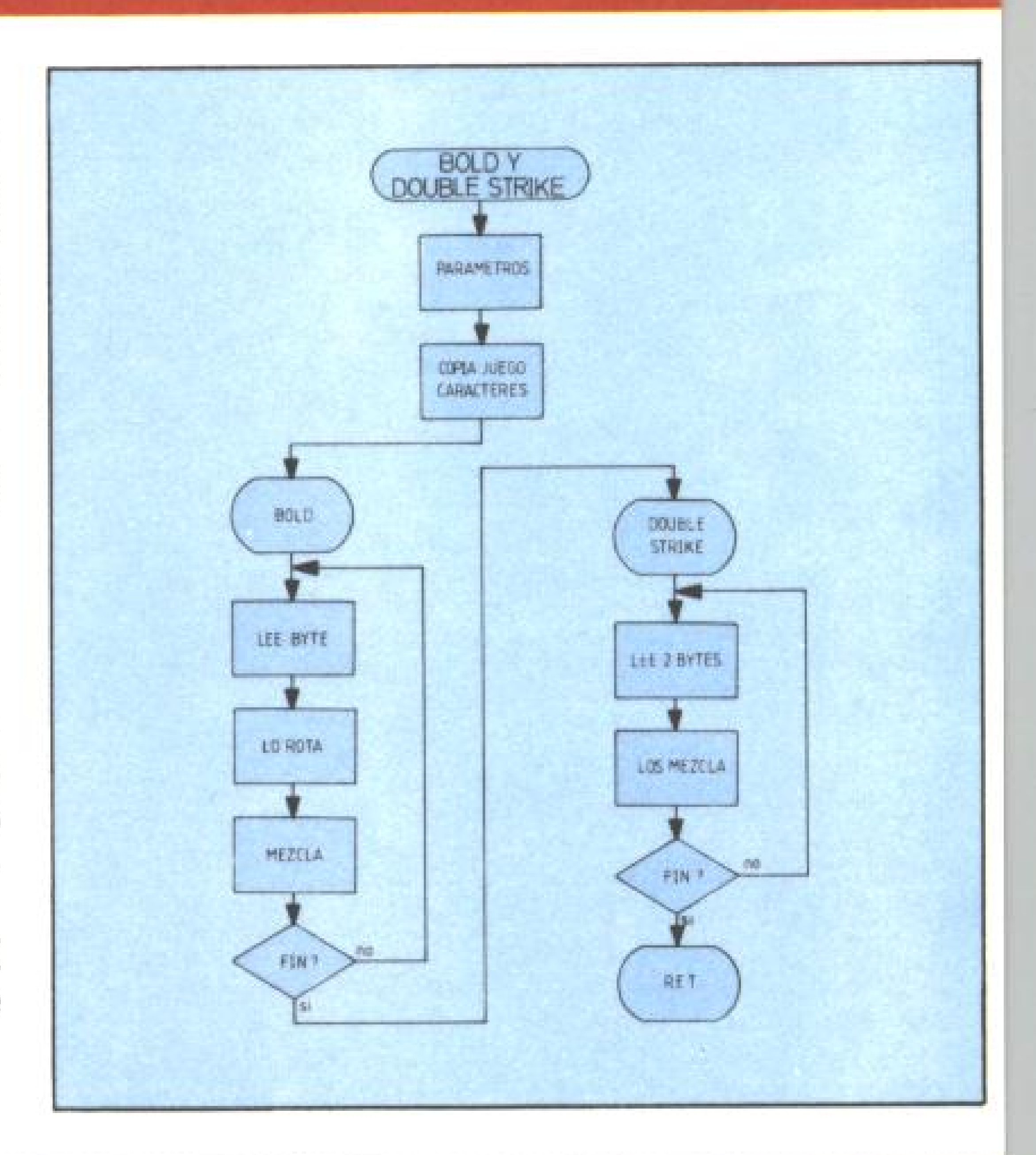
Aprovechando que el valor de CHARS puede variarse, podemos crear un nuevo juego de caracteres localizado en RAM, consistente en letras de doble grosor, tanto en el ancho (BOLD) como en alto (DOUBLE STRIKE). Este juego se almacena entre las direcciones 61000 y 62023.

La rutina se utiliza con RANDOMIZE USR N, siendo N la dirección donde se ubicará la rutina.

Funcionamiento:

La rutina comienza copiando en la RAM la tabla de caracteres de la ROM, posteriormente crea el tipo de letra BOLD (BUCLEA) mezclando los 8 bits de cada caracter con los de su derecha.

Para crear el tipo de letra DOUBLE STRIKE (BUCLEB) mezcla cada byte con el que tiene debajo (sólo en las mayúsculas).



```
 10 ;  **BOLD Y DOUBLE STRIKE **
 20 ;
 30 ;
 40         ORG     60000    ;RUTINA REUBICABLE
 50 ;
 60         LD      HL,(CHARS)
 70         LD      DE,NJUEGO;Nuevo juego de CHRs
 80         LD      (CHARS),DE;Nuevo valor CHARS
 90         LD      BC,1024  ;Long. del juego
100         LDIR    ;        ;Copia el juego de
110 ;                        ;  caracteres
120 ;
130 ;
140 ; LETRAS BOLD
150 ;
160 START1  LD      HL,NJUEGO
170         LD      BC,1024  ;Todos los caracteres
180 BUCLEA  LD      A,(HL)   ;Lee un byte
190         RR      A        ; lo rota a la dere.
200         OR      (HL)     ; y lo mezcla
210         LD      (HL),A   ;  consigo mismo
220         INC     HL
230         DEC     BC       ;Contador de bytes
240         LD      A,B
250         OR      C        ;Comprueba si BC=0
260         JR      NZ,BUCLEA; si no repite bucle
270 ;
280 ;
290 ;  DOUBLE STRIKE
300 ;
310 START2  LD      HL,NJUEGO+520;Dir. de la A
320         LD      BC,208   ;Solo las mayusculas
330 BUCLEB  LD      A,(HL)   ;Lee un byte
340         INC     HL       ;  lo mezcla
350         OR      (HL)     ;  con el que
360         DEC     HL       ;  tiene debajo
370         LD      (HL),A   ;  y lo guarda
380 ;                        ;  en el de arriba
390         INC     HL
400         DEC     BC       ;Contador de bytes
410         LD      A,B
420         OR      C
430         RET     Z        ;Retorna si BC=0
440         JR      BUCLEB
450 ;
460 ;
470 CHARS   EQU     23606    ;Dir. tabla caract.
480 NJUEGO  EQU     61000    ;Nuevo juego de cars.
```

```
10 DATA "2A 36 5C 11 48 EE ED 53",835
20 DATA "36 5C 01 00 04 ED B0 21",597
30 DATA "48 EE 01 00 04 7E CB 1F",675
40 DATA "B6 77 23 0B 78 B1 20 F5",921
50 DATA "21 50 F0 01 D0 00 7E 23",723
60 DATA "B6 2B 77 23 0B 78 B1 C8",887
70 DATA "18 F4                   ",268
```

Lógica

Podemos realizar las funciones lógicas elementales AND, OR y XOR, de una forma binaria, con números de 16 bits.

Su uso debe ser:

«LET resultado = I + J * K ↑ USR nn»

donde I, J y K son operandos que se detallan en la tabla siguiente, y nn es la dirección de comienzo de la rutina.

Valor de K	Funcion realizada
0	I AND J
1	I OR J
otros	I XOR J

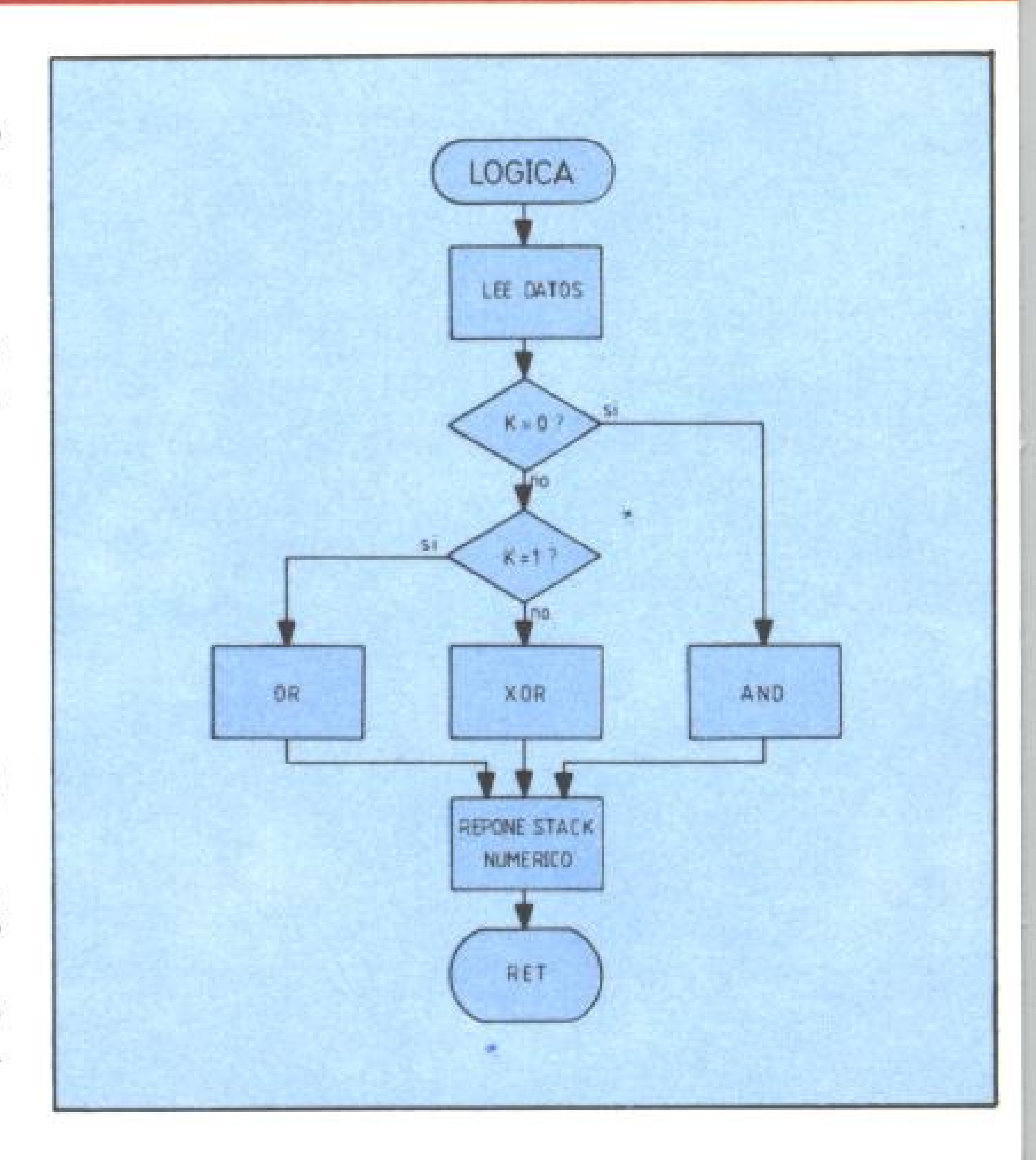
Funcionamiento:

Tres llamadas consecutivas a la ROM (FINT 2) se utilizan para tomar los valores de I, J y K.

El valor de K, determina la función a realizar.

La correspondiente rutina efectúa dos veces la función, una para cada byte.

La rutina FIN restablece el stack numérico (STKBC) de modo que el resultado de la operación sea el adecuado.



```
 10 ; * LOGICA *
 20 ;
 30         ORG     60000    ;RUTINA REUBICABLE
 40         CALL    FINT1    ;Lee K del STK
 50         PUSH    AF
 60         CALL    FINT2    ;Lee J del STK
 70         PUSH    BC
 80         CALL    FINT2    ;Lee I del STK
 90         PUSH    BC
100         POP     HL       ;Transfiere I a HL
110         POP     DE       ;Transfiere J a DE
120         POP     AF       ;Transfiere K a A
130         AND     A
140         JR      Z,BAND   ;Si es 0 realiza AND
150         DEC     A
160         JR      Z,BOR    ;Si es 1 realiza OR
170 ;                        ;Otro valor hace XOR
180 ;
190 BXOR    LD      A,E      ;Realiza I XOR J
200         XOR     L
210         LD      C,A      ;C=E XOR L
220         LD      A,D
230         XOR     H
240         LD      B,A      ;B=D XOR H
250         JR      FIN
260 ;
270 BAND    LD      A,E      ;Realiza I AND J
280         AND     L
290         LD      C,A      ;C=E AND L
300         LD      A,D
310         AND     H
320         LD      B,A      ;B=D AND H
330         JR      FIN
340 ;
350 BOR     LD      A,E      ;Realiza I OR J
360         OR      L
370         LD      C,A      ;C=E OR L
380         LD      A,D
390         OR      H
400         LD      B,A      ;B=D OR H
410 ;
420 FIN     CALL    STKBC    ;I=resultado en STK
430         LD      BC,0
440         CALL    STKBC    ;J=0 en STK
450         LD      BC,1
460         PUSH    BC
470         CALL    STKBC    ;K=1 en STK
480         POP     BC       ;Valor del USR=1
490         RET     ;        ;I+0*1^1=I
500 FINT1   EQU     #1E94    ;Lee en A el STK num.
510 FINT2   EQU     #1E99    ;Lee en BC el STK nu.
520 STKBC   EQU     #2D2B    ;Guar. BC en STK num.
```

```
10 DATA "CD 94 1E F5 CD 99 1E C5",1213
20 DATA "CD 99 1E C5 E1 D1 F1 A7",1427
30 DATA "28 0B 3D 28 10 7B AD 4F",543
40 DATA "7A AC 47 18 0E 7B A5 4F",770
50 DATA "7A A4 47 18 06 7B B5 4F",770
60 DATA "7A B4 47 CD 2B 2D 01 00",667
70 DATA "00 CD 2B 2D 01 01 00 C5",492
80 DATA "CD 2B 2D C1 C9          ",687
```

SCROLL vertical

Si queremos producir un desplazamiento hacia arriba de la pantalla basta con hacer una llamada a la rutina de la ROM de la forma:

RANDOMIZE USR 3582

Para desplazar la pantalla hacia abajo se deberá usar esta rutina mediante:

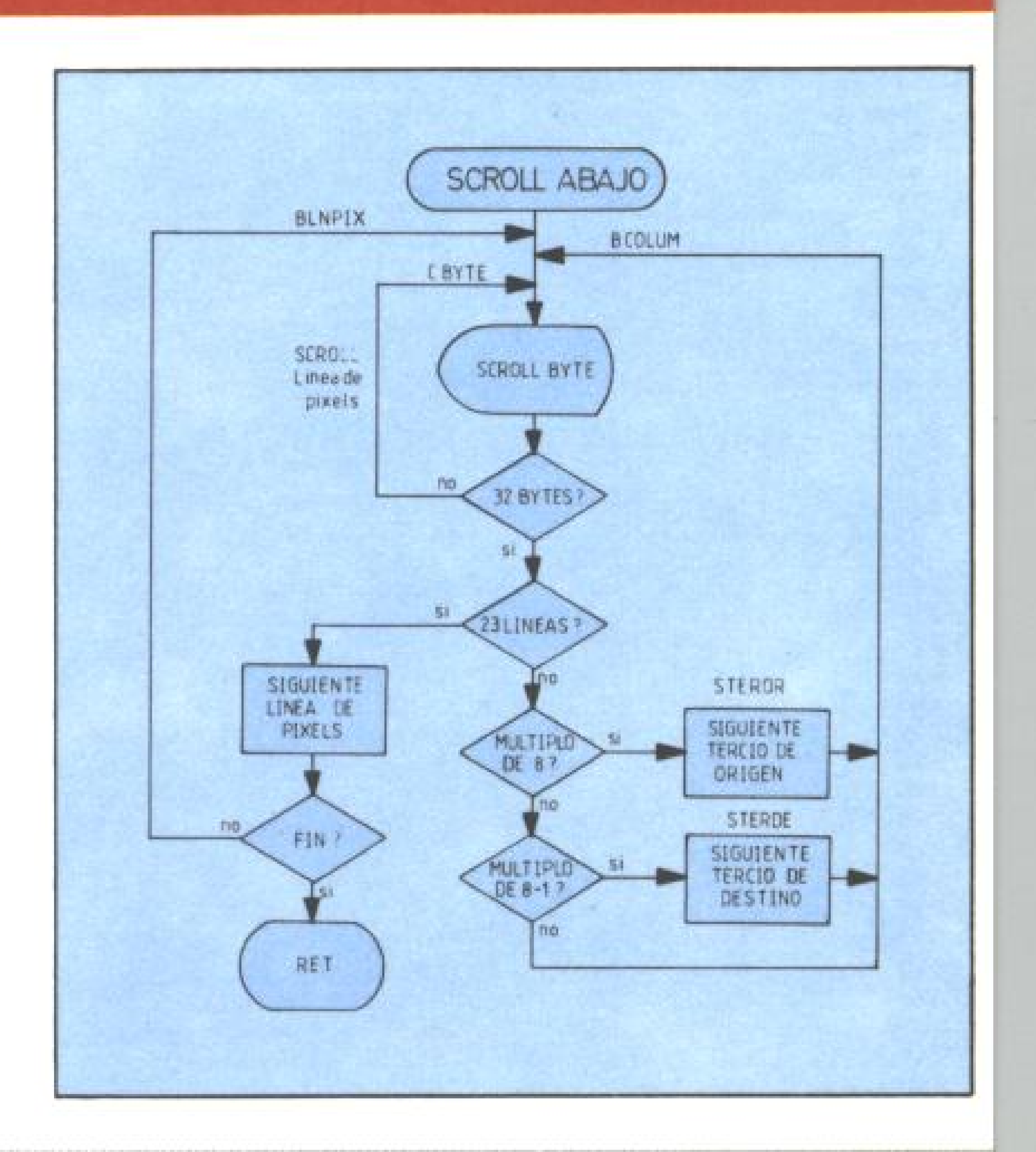
RANDOMIZE USR N

donde N es la dirección en la que se encuentre la rutina (es reubicable).

Para hacer el scroll de los atributos deberá utilizarse la rutina correspondiente de la ficha «SCROLL DE ATRIBUTOS».

Funcionamiento:

Va desplazando hacia abajo primero el octavo byte de todos los caracteres, después el séptimo, y así sucesivamente, hasta hacerlo con toda la pantalla. Cada vez que llega al final de un tercio, se dirige a las subrutinas STEROR (sig. tercio de origen) y STERDE (sig. tercio de destino) que calculan las direcciones correspondientes al siguiente tercio.



```
 10 ;** SCROLL ABAJO **
 20         ORG     60000    ;RUTINA REUBICABLE
 30 ;
 40 START   LD      DE,22527 ;Ultimo byte linea 23
 50         LD      HL,22495 ;Ultimo byte linea 22
 60 ;
 70 BLNPIX  PUSH    HL
 80         PUSH    DE
 90         LD      C,23     ;No. de lineas-1
100 BCOLUM  LD      B,32     ;No. de columnas
110 CBYTE   LD      A,(HL)   ;A=byte a copiar
120         LD      (DE),A   ;  lo copia
130         XOR     A        ;Borra el antiguo
140         LD      (HL),A   ;  byte de origen
150         DEC     HL       ;Sig. byte origen
160         DEC     DE       ;Sig. byte destino
170         DJNZ    CBYTE    ;Bucle linea pixels
180         DEC     C        ;Contador de lineas
190         JR      Z,CLNPIX ;Si linea=0
200 ;                        ;  sig. linea pixels
210         LD      A,C
220         AND     7
230         CP      0        ;Pasa al sig. tercio
240         JR      Z,STEROR ;  de origen
250         CP      7        ;Pasa al sig. tercio
260         JR      Z,STERDE ;  de destino
270         JR      BCOLUM
280 ;
290 STEROR  PUSH    DE       ;Guar. puntero dest.
300         LD      DE,1792  ;Dist. al sig. terc.
310         XOR     A        ;Carry a 0
320         SBC     HL,DE    ;HL=sig. tercio
330         POP     DE       ;Rec. puntero destino
340         JR      BCOLUM   ;Bucle de columnas
350 ;
360 STERDE  PUSH    HL       ;Guar. puntero orig.
370         EX      DE,HL
380         LD      DE,1792  ;Dist. al sig. tercio
390         XOR     A        ;Carry a 0
400         SBC     HL,DE    ;HL=sig. tercio
410         EX      DE,HL    ;DE=sig. tercio
420         POP     HL       ;Rec. puntero origen
430         JR      BCOLUM   ;Bucle de columnas
440 ;
450 CLNPIX  POP     DE       ;Guar. puntero dest.
460         POP     HL       ;Guar. puntero origen
470         DEC     D        ;Sig. linea de pixels
480         DEC     H        ;Sig. linea de pixels
490         LD      A,D
500         CP      79       ;Si no ha acabado la
510         JR      NZ,BLNPIX;  sig. linea
520         RET
```

```
10 DATA "11 FF 57 21 DF 57 E5 D5",1144
20 DATA "0E 17 06 20 7E 12 AF 77",513
30 DATA "2B 1B 10 F8 0D 28 23 79",543
40 DATA "E6 07 FE 00 28 06 FE 07",798
50 DATA "28 0C 18 E6 D5 11 00 07",543
60 DATA "AF ED 52 D1 18 DC E5 EB",1411
70 DATA "11 00 07 AF ED 52 EB E1",978
80 DATA "18 D0 D1 E1 15 25 7A FE",1100
90 DATA "4F 20 C3 C9             ",507
```

SCROLL horizontal

Estas dos rutinas independientes entre si y reubicables ofrecen la posibilidad de hacer un desplazamiento del DISPLAY FILE de un carácter a derecha o izquierda.

Su forma de llamada es:

RANDOMIZE USR N

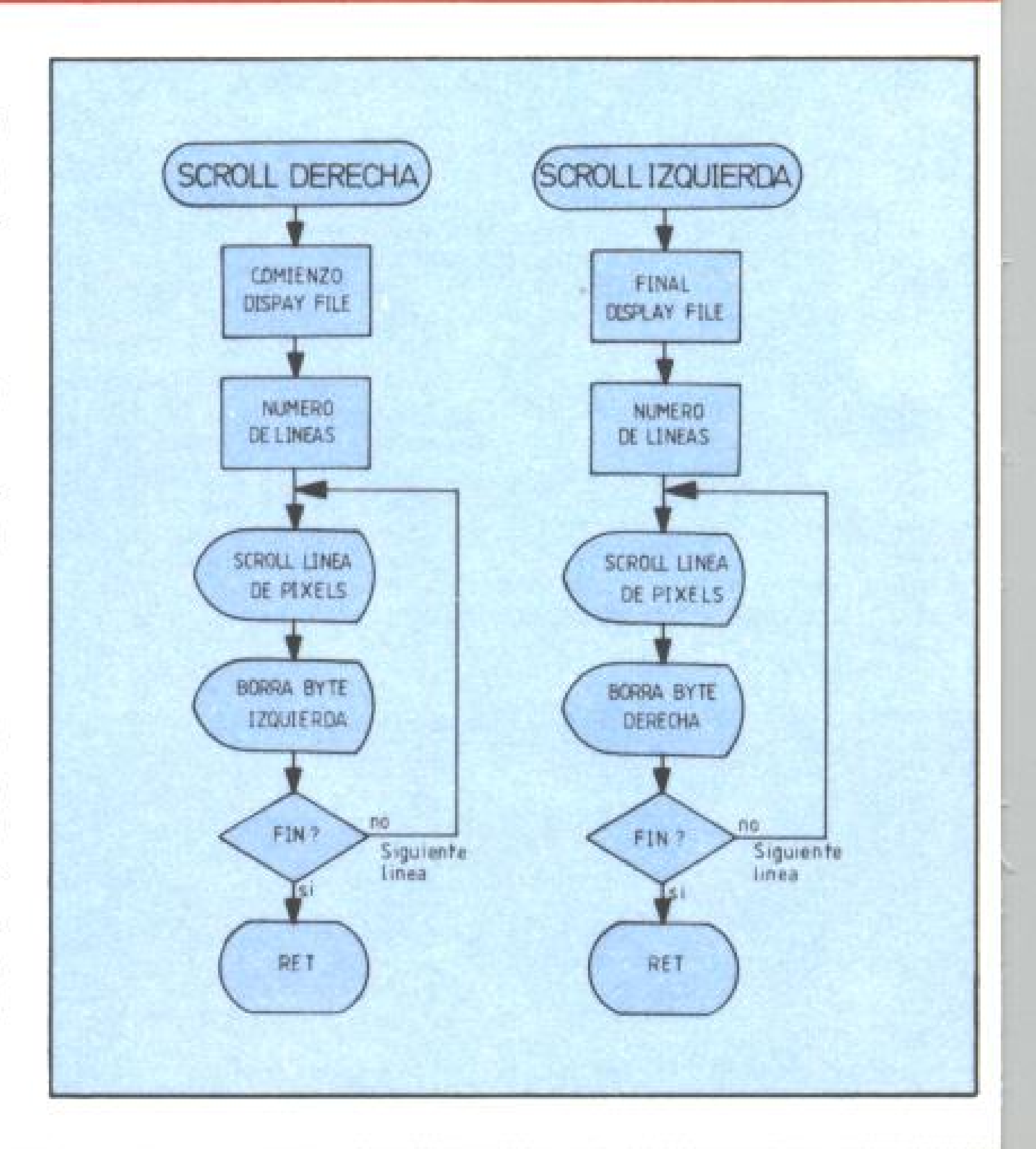
siendo N la dirección donde haya sido ubicada cada una.

Para desplazar los correspondientes atributos deberán utilizarse las rutinas de la ficha «Scroll de atributos».

Funcionamiento:

Constan de un bucle de 64 * T vueltas (T es el número de tercios de pantalla a desplazar) en que las instrucciones LDDR y LDIR desplazan 31 bytes y LD (DE),A borra el byte sobrante; «A» fue puesto a 0 mediante la instrucción XOR A.

El Scroll a la derecha comienza por el último byte del DISPLAY FILE y el de la izquierda por el primero.



```
 10 ;** SCROLL IZQUIERDA EN BAJA RESOLUCION **
 20 ;
 30        ORG    60000    ;RUTINA REUBICABLE
 40 ;
 50 START  LD     DE,16384 ;Comienzo DISP. FILE
 60        LD     HL,16385 ;Sig. byte
 70        LD     B,64*3   ;3 tercios de 64
 80 ;                      ;  lineas cada uno
 90        XOR    A        ;A=0
100 ;
110 SBICOL PUSH   BC       ;Guar. no. de lineas
120        LD     BC,31    ;31 columnas
130        LDIR   ;        ;Mueve linea pixels
140        LD     (DE),A   ;Borra byte derecha
150        INC    HL       ;Sig. linea de origen
160        INC    DE       ;Sig. lin. de destino
170        POP    BC       ;Recupera contador
180 ;                      ;  de lineas
190        DJNZ   SBICOL   ;Scroll sig. linea
200        RET
```

```
10 DATA "11 00 40 21 01 40 06 C0",377
20 DATA "AF C5 01 1F 00 ED B0 12",835
30 DATA "23 13 C1 10 F4 C9      ",708
```

```
 10 ;** SCROLL DERECHA EN BAJA RESOLUCION **
 20 ;
 30        ORG    60000    ;RUTINA REUBICABLE
 40 ;
 50 START  LD     DE,22527 ;Fin del DISPLAY FILE
 60        LD     HL,22526 ;Un byte menos
 70        LD     B,64*3   ;3 tercios de 64
 80 ;                      ;  lineas cada uno
 90        XOR    A        ;A=0
100 ;
110 SBDCOL PUSH   BC       ;Guar. no. de lineas
120        LD     BC,31    ;31 columnas
130        LDDR   ;        ;Mueve linea pixels
140        LD     (DE),A   ;Borra byte izquierda
150        DEC    DE       ;Sig. lin. de destino
160        DEC    HL       ;Sig. linea de origen
170        POP    BC       ;Recupera contador
180 ;                      ;  de lineas
190        DJNZ   SBDCOL   ;Scroll sig. linea
200        RET
```

```
10 DATA "11 FF 57 21 FE 57 06 C0",931
20 DATA "AF C5 01 1F 00 ED B8 12",843
30 DATA "1B 2B C1 10 F4 C9      ",724
```

SCROLL de atributos

Ofrecemos cuatro rutinas de scroll únicamente de atributos.

Las cuatro rutinas son independientes y su forma de utilización es:

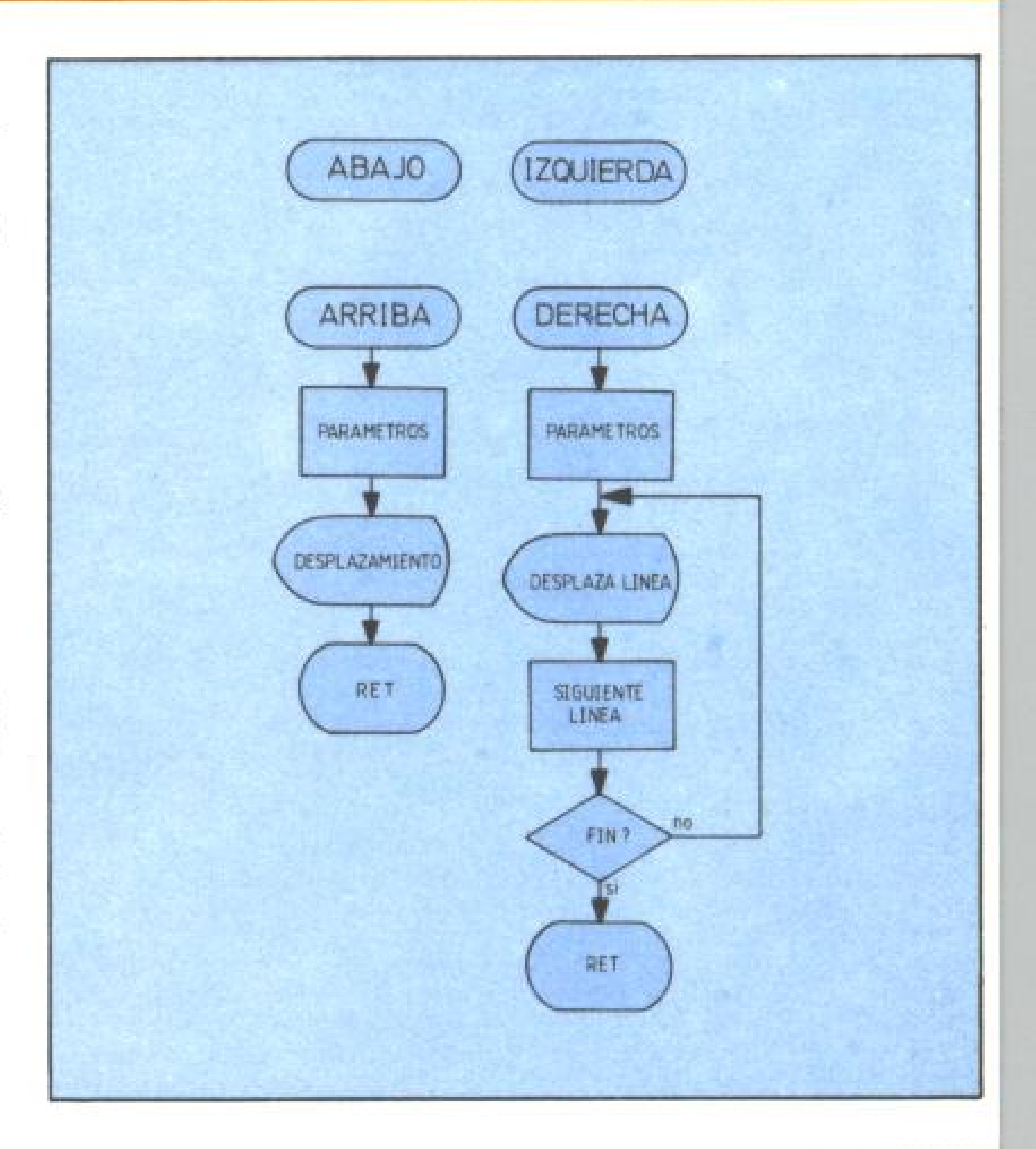
RANDOMIZE USR N . Scroll abajo.
RANDOMIZE USR N+12 . Scroll arriba.
RANDOMIZE USR N+24 . Scroll derecha.
RANDOMIZE USR N+48 . Scroll izquierda.

Donde N será la dirección en que se ubique la rutina.

Funcionamiento:

Las rutinas de scroll arriba y abajo desplazan con un LDDR (scroll abajo) o un LDIR (scroll arriba) el fichero de atributos.

Las de scroll a derecha e izquierda van recorriendo línea por línea toda la pantalla desplazándolas con LDDR o LDIR en uno u otro sentido.



```
 10 ;  ** RUTINAS DE SCROLL DE ATRIBUTOS **
 20 ;
 30        ORG    60000     ;RUTINAS REUBICABLES
 40 ;
 50 ;  SCROLL DE ATRIBUTOS ABAJO
 60 ;
 70 START1 LD     DE,DBATR+767;Linea 23
 80        LD     HL,DBATR+735;Linea 22
 90        LD     BC,736    ;736 caracteres
100        LDDR
110        RET
120 ;
130 ;  SCROLL DE ATRIBUTOS ARIBA
140 ;
150 START2 LD     HL,DBATR+32;Linea 1
160        LD     DE,DBATR  ;Linea 0
170        LD     BC,672    ;672 caracteres
180        LDIR
190        RET
200 ;
210 ;  SCROLL DE ATRIBUTOS A LA DERECHA
220 ;
230 START3 LD     HL,DBATR+30;Penultima columna
240        LD     DE,DBATR+31;Ultima columna
250        LD     A,22      ;Lin. de la pantalla
260 XSD1   LD     BC,31     ;31 columnas
270        LDDR   ;         ;Desplaza a la der.
280        LD     BC,64     ;Dist. a la sig. lin.
290        ADD    HL,BC     ;HL=Sig. linea
300        LD     D,H
310        LD     E,L       ;DE=HL
320        DEC    HL        ;Un caracter atras
330        DEC    A         ;Contador de lineas
340        JR     NZ,XSD1   ;Si A<>0 repite bucle
350        RET
360 ;
370 ;  SCROLL DE ATRIBUTOS A LA IZQUIERDA
380 ;
390 START4 LD     HL,DBATR+1;Segunda columna
400        LD     DE,DBATR  ;Primera columna
410        LD     A,22      ;Lin. de la pantalla
420 XSI1   LD     BC,31     ;31 columnas
430        LDIR   ;         ;Desp. a la izq.
440        INC    HL        ;Un caracter adelante
450        INC    DE        ;Car. adelante dest.
460        DEC    A         ;Contador de lineas
470        JR     NZ,XSI1   ;Si A<>0 repite bucle
480        RET
490 DBATR  EQU    22528
```

```
10 DATA "11 FF 5A 21 DF 5A 01 E0",933
20 DATA "02 ED B8 C9 21 20 58 11",794
30 DATA "00 58 01 A0 02 ED B0 C9",865
40 DATA "21 1E 58 11 1F 58 3E 16",371
50 DATA "01 1F 00 ED B8 01 40 00",518
60 DATA "09 54 5D 2B 3D 20 F1 C9",764
70 DATA "21 01 58 11 00 58 3E 16",311
80 DATA "01 1F 00 ED B0 23 13 3D",560
90 DATA "20 F6 C9                ",479
```

SCROLL derecha

Realiza un scroll en baja resolución hacia la derecha de toda la pantalla, incluidos los atributos. La parte de la izquierda se borra recibiendo el color de atributos permanentes.

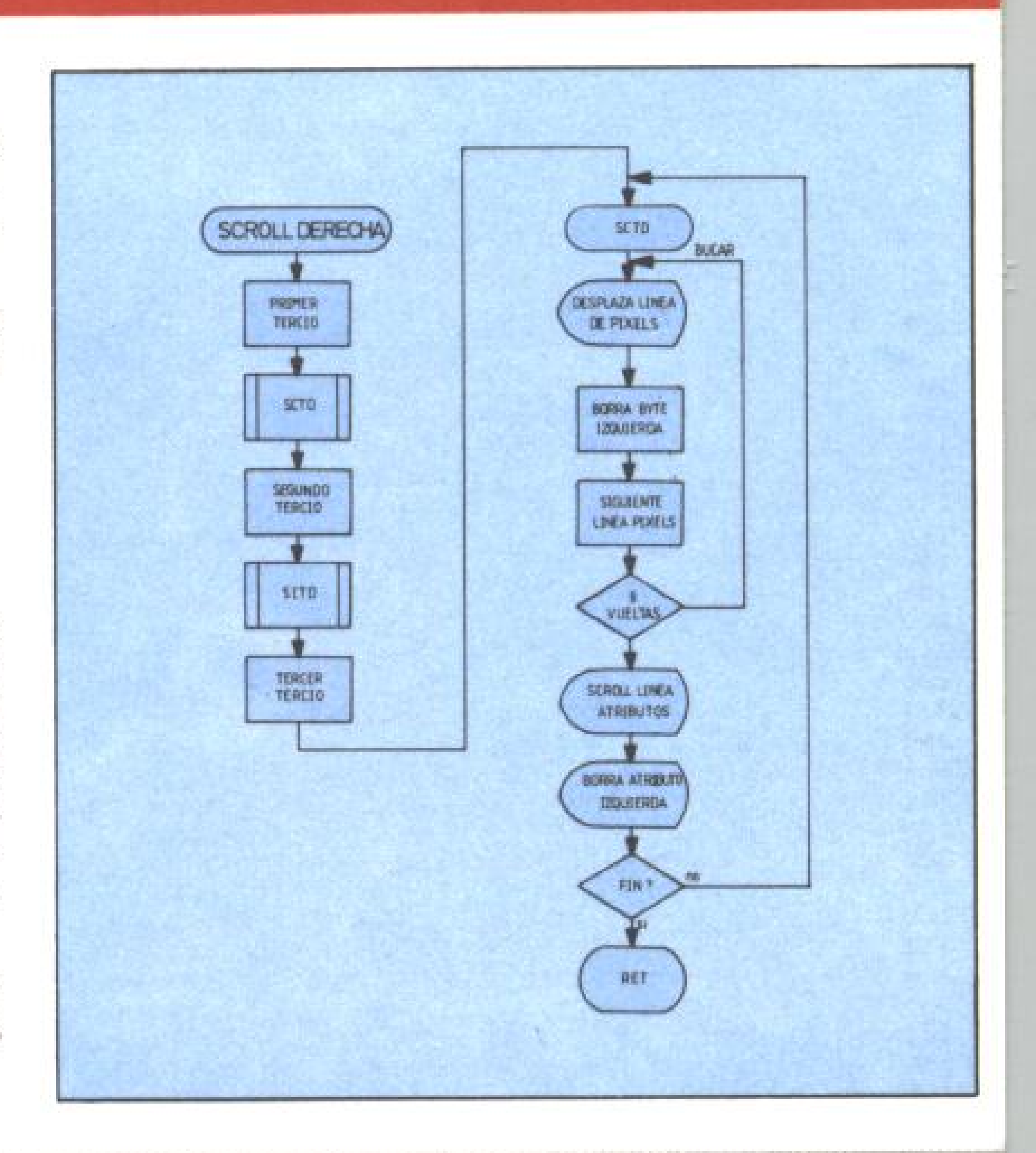
La rutina no es reubicable y está localizada en la dirección 60100. Para producir el scroll debe hacerse:

RANDOMIZE USR 60100

Funcionamiento:

Consiste en tres llamadas a la subrutina SCTD, una para cada tercio de la pantalla. En esta, se desplazan hacia la dercha (primero el DISPLAY FILE y después el ATTRIBUTE FILE) el número de lineas indicado por el acumulador A (inicialmente 8). Modificando este valor podemos conseguir que el scroll sólo efecte al número de líneas que se desee en cada tercio.

La rutina SCTD consta de dos bucles anidados, el interior (BUCAR) mueve líneas de pixels y el exterior las de caracteres.



```
 10 ; ** SCROLL A LA DERECHA **
 20        ORG    60100     ;RUTINA NO REUBICABLE
 30 START  LD     HL,#5800  ;Dir. com. de atrib.
 40        LD     (DATTR),HL;  lo guarda
 50        LD     DE,#401F  ;Primer tercio
 60        LD     HL,#401E  ;  de la pantalla
 70        LD     A,8       ;Tercio completo
 80        CALL   SCTD      ;Scroll del tercio
 90        LD     DE,#481F  ;Segundo tercio
100        LD     HL,#481E  ;  de la pantalla+31
110        LD     A,8       ;Tercio completo
120        CALL   SCTD      ;Scroll del tercio
130        LD     DE,#501F  ;Tercer tercio
140        LD     HL,#501E
150        LD     A,8       ;Tercio completo
160 SCTD   PUSH   AF        ;Guar. num. de lineas
170        LD     A,8       ;8 lineas de pixels
180 BUCAR  LD     BC,31     ;Scroll de 31
190        LDDR   ;         ;  columnas
200        INC    HL        ;El byte ultimo
210        LD     (HL),0    ;  lo borra
220        LD     BC,287    ;Dist. a la siguiente
230        ADD    HL,BC     ;  linea de pixels
240        LD     D,H       ;
250        LD     E,L       ;DE=HL
260        DEC    HL        ;Sig. linea pixels
270        DEC    A         ;Contador de lineas
280        JR     NZ,BUCAR  ;Scroll sig. linea
290        PUSH   HL        ;Puntero DISP.FILE
300        LD     HL,(DATTR);Recu. dir. ATTR
310        LD     BC,31     ;Scroll de 31 colum.
320        ADD    HL,BC     ;Prx. lin. de caract.
330        PUSH   HL        ;Puntero de atributos
340        INC    HL
350        LD     (DATTR),HL;Guarda dir. sig.
360        DEC    HL        ;  linea de atributos
370        DEC    HL        ;Scroll a la
380        POP    DE        ;  derecha de la
390        LDDR   ;         ;  linea de atributos
400        INC    HL
410        LD     A,(23693);ATTR de pantalla
420        LD     (HL),A    ;Borra atributos
430        POP    HL        ;Rec. dir. DISP.FILE
440        LD     BC,2015   ;Long. tercio-33
450        SBC    HL,BC     ;Prox. linea de
460        LD     D,H       ;  caracteres
470        LD     E,L       ;DE=HL
480        DEC    HL        ;Un caracter atras
490        POP    AF        ;Recupera no. lineas
500        DEC    A         ;Otra linea
510        JR     NZ,SCTD   ;Scroll linea sig.
520        RET
530 DATTR  DEFW   #5800     ;Memoria auxiliar
```

```
 10 DATA "21 00 58 22 22 EB 11 1F",472
 20 DATA "40 21 1E 40 3E 08 CD E8",698
 30 DATA "EA 11 1F 48 21 1E 48 3E",551
 40 DATA "08 CD E8 EA 11 1F 50 21",840
 50 DATA "1E 50 3E 08 F5 3E 08 01",496
 60 DATA "1F 00 ED B8 23 36 00 01",542
 70 DATA "1F 01 09 54 5D 2B 3D 20",354
 80 DATA "EE E5 2A 22 EB 01 1F 00",810
 90 DATA "09 E5 23 22 22 EB 2B 2B",662
100 DATA "D1 ED B8 23 3A 8D 5C 77",1075
110 DATA "E1 01 DF 07 ED 42 54 5D",936
120 DATA "2B F1 3D 20 C7 C9 00 58",865
130 DATA "                        ",0
```

SCROLL izquierda

Dentro de la serie de rutinas de scroll, ésta produce un desplazamiento de un carácter hacia la izquierda de toda la pantalla, incluidos los atributos. La parte de la derecha es borrada y recibe el color de atributos permanentes.

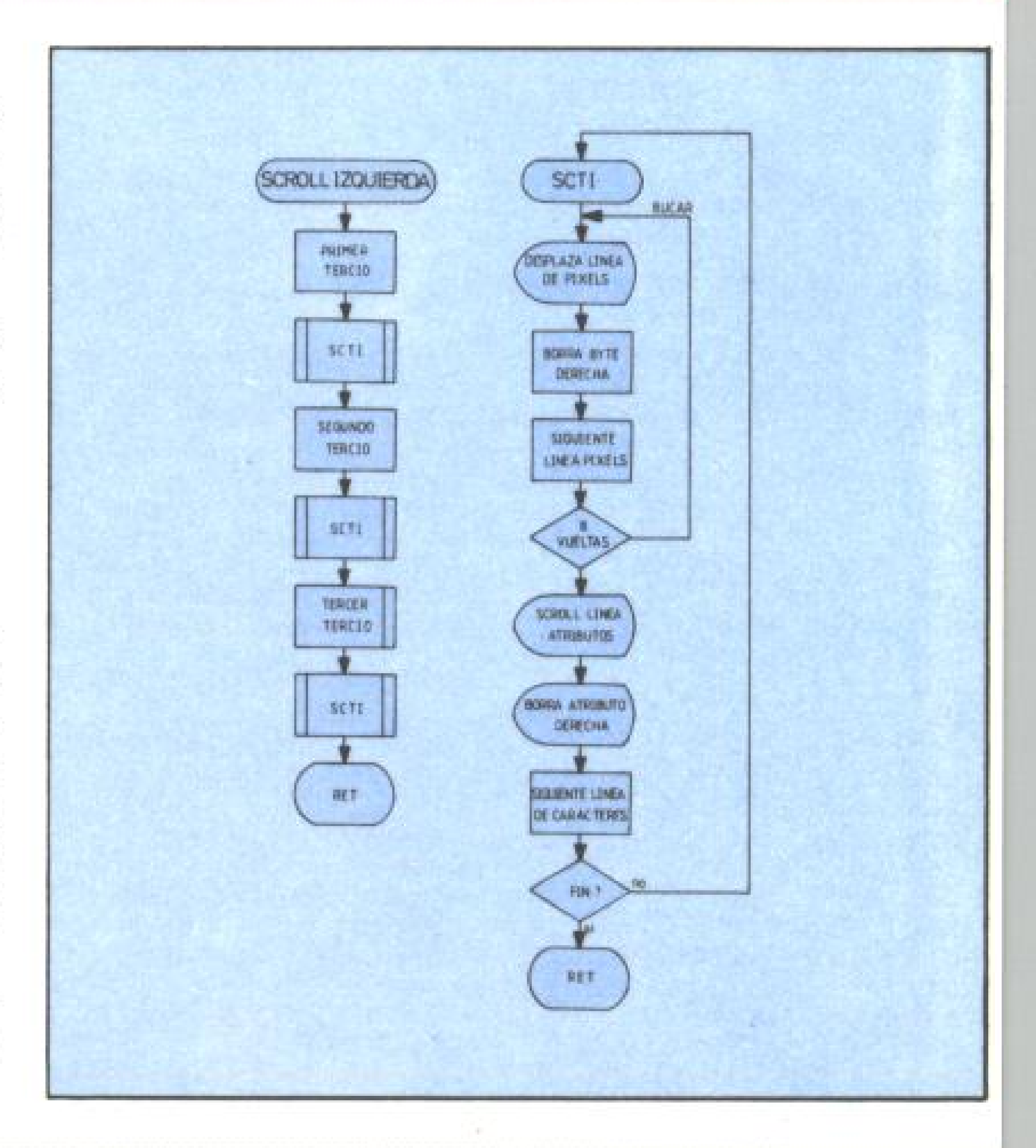
La rutina no es reubicable. se localiza en la dirección 60200. Para producir el scroll se hará:

RANDOMIZE USR 60200

Funcionamiento:

Se efectúan tres llamadas a la subrutina SCTI, una por cada tercio de la pantalla. En esta se desplazan hacia la izquierda (primero el DISPLAY FILE y después el ATTRIBUTE FILE) el número de líneas indicado por el acumulador A (inicialmente 8). Modificando su valor conseguiremos que el scroll sólo afecte al número de líneas que deseemos para cada tercio.

La rutina SCTI consta de dos bucles anidados, el menor (BUCAR) mueve líneas de pixels y el mayor, líneas de caracteres.



```
10 ; ** SCROLL A LA IZQUIERDA **
20        ORG    60200     ;RUTINA NO REUBICABLE
30 START  LD     HL,#5800  ;Comzo. de atributos
40        LD     (DATTR),HL;  lo guarda
50        LD     DE,#4000  ;Primer tercio
60        LD     HL,#4001  ;  de la pantalla
70        LD     A,8       ;Tercio completo
80        CALL   SCTI      ;Scroll del tercio
90        LD     DE,#4800  ;Segundo tercio
100       LD     HL,#4801
110       LD     A,8       ;Tercio completo
120       CALL   SCTI      ;Scroll del tercio
130       LD     DE,#5000  ;Tercer tercio
140       LD     HL,#5001
150       LD     A,8       ;Tercio completo
160       CALL   SCTI      ;Scroll del tercio
170       RET    ;         ; Fin
180 SCTI  PUSH   AF        ;Guar. num. de lineas
190       LD     A,8       ;8 lineas de pixels
200 BUCAR LD     BC,31     ;Scroll de 31
210       LDIR   ;         ;  columnas
220       DEC    HL        ;El byte ultimo
230       LD     (HL),0    ;  lo borra
240       LD     BC,225    ;Dist. a la siguiente
250       ADD    HL,BC     ;  linea de pixels
260       LD     D,H       ;
270       LD     E,L       ;DE=HL
280       INC    HL        ;Segundo pixel
290       DEC    A         ;Contador de lineas
300       JR     NZ,BUCAR  ;Scroll sig. linea
310       PUSH   HL        ;Puntero DISP.FILE
320       LD     HL,(DATTR);Recup. dir. ATTR
330       LD     D,H       ;
340       LD     E,L       ;DE=HL
350       INC    HL        ;Scroll de
360       LD     BC,31     ;  31 caracteres
370       LDIR   ;         ;de atributos
380       LD     (DATTR),HL;Guarda dir. sig.
390       DEC    HL        ;  linea de atributos
400       LD     A,(23693);ATTR de pantalla
410       LD     (HL),A    ;Borra atributo
420       POP    HL        ;Rec. dir. DISP.FILE
430       LD     BC,2016   ;Long. tercio-32
440       SBC    HL,BC     ;Prox. linea de
450       LD     D,H       ;  caracteres
460       LD     E,L       ;DE=HL
470       DEC    DE        ;Un caracter atras
480       POP    AF        ;Recupera no. lineas
490       DEC    A         ;Otra linea
500       JR     NZ,SCTI   ;Scroll linea sig.
510       RET
520 DATTR DEFW   #5800     ;Memoria auxiliar
```

```
 10 DATA "21 00 58 22 87 EB 11 00",542
 20 DATA "40 21 01 40 3E 08 CD 50",517
 30 DATA "EB 11 00 48 21 01 48 3E",492
 40 DATA "08 CD 50 EB 11 00 50 21",658
 50 DATA "01 50 3E 08 CD 50 EB C9",872
 60 DATA "F5 3E 08 01 1F 00 ED B0",760
 70 DATA "2B 36 00 01 E1 00 09 54",416
 80 DATA "5D 23 3D 20 EE E5 2A 87",865
 90 DATA "EB 54 5D 23 01 1F 00 ED",716
100 DATA "B0 22 87 EB 2B 3A 8D 5C",914
110 DATA "77 E1 01 E0 07 ED 42 54",963
120 DATA "5D 1B F1 3D 20 CA C9 00",857
130 DATA "58                     ",88
```

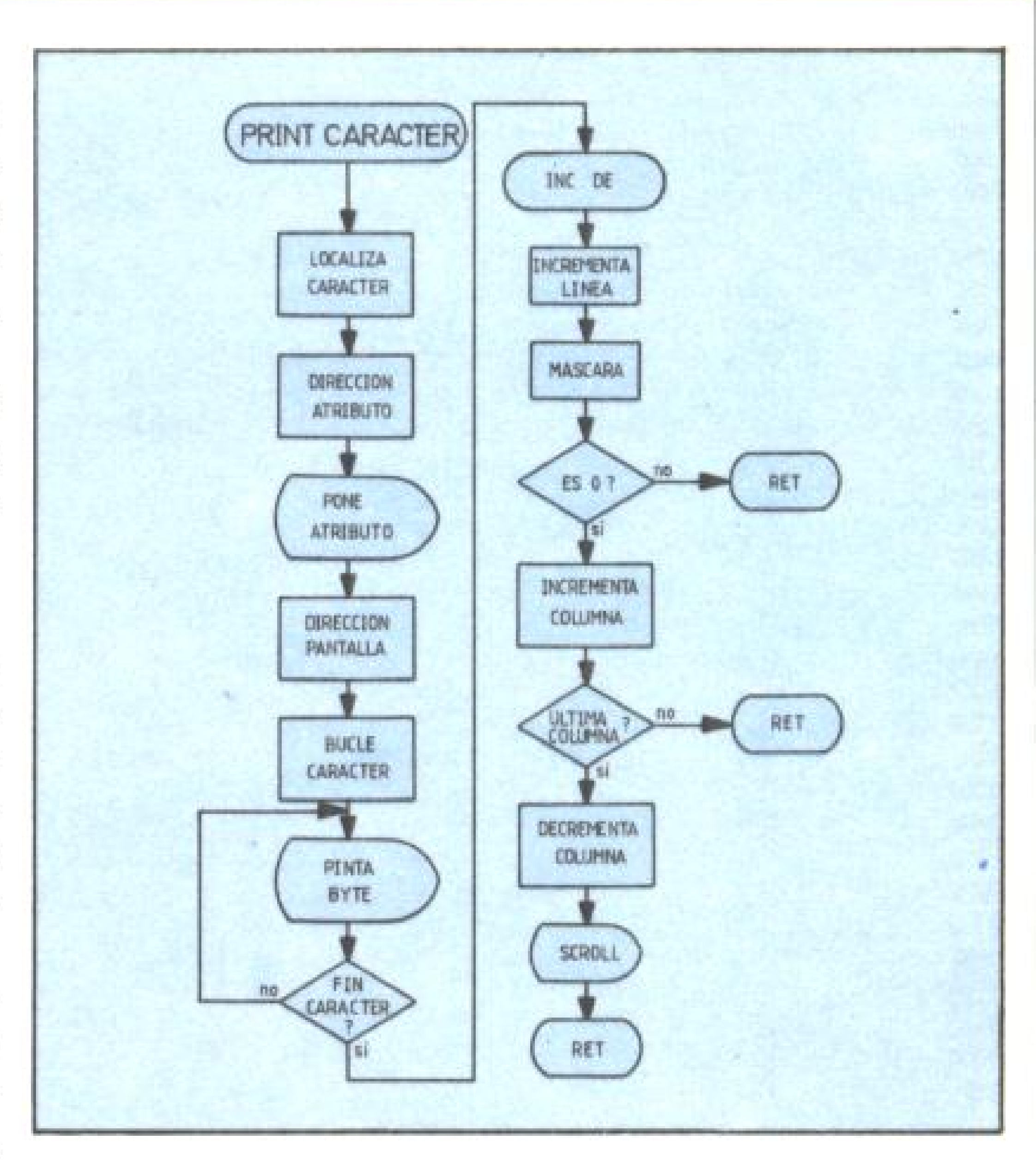
Sustituye a la llamada RST 10H para impresión de un caracter en pantalla con una velocidad mucho mayor y permitiendo una tabla de 256 caracteres. La rutina sólo es útil desde código máquina y la forma de llamada es CALL START.

Antes de hacer la llamada, HL debe contener el código del caracter; A, el atributo; D, la línea y E, la columna. A su retorno habrá incrementado el puntero DE.

Funcionamiento:

La primera parte calcula la dirección de comienzo del caracter, que guarda en HL, posteriormente halla la dirección en el archivo, de atributos, donde asigna A. Por último calcula la posición en el display file, recupera el comienzo del caracter y lo dibuja mediante el bucle «BUPINT».

La segunda parte incrementa las coordenadas (INCDE). En caso de encontrarse en el último caracter de la pantalla hace un scroll y sitúa el puntero al comienzo de la última línea.



```
10 ;        *  PRINT UN CARACTER *
20 ;   H-> 0        L-> CARACTER
30 ;   D-> Linea  C->Columna
40 ;   A-> Atributos
50 ;
60 ;
70 PRINT   LD      BC,(CHARS);Comzo. tabla cars.
80         ADD     HL,HL     ;HL=HL*2
90         ADD     HL,HL     ;HL=HL*4
100        ADD     HL,HL     ;HL=HL*8
110        ADD     HL,BC     ;HL=Dir. del caracter
120        PUSH    HL        ;Guarda dir. caracter
130        LD      L,D       ;L=Linea (Y)
140        LD      H,0
150        ADD     HL,HL     ;La dir. en ATTR FILE
160        ADD     HL,HL     ;  es #5800+D*32+E
170        ADD     HL,HL
180        ADD     HL,HL     ;  D*32
190        ADD     HL,HL
200        LD      B,#58     ;Byte alto del A.FILE
210        LD      C,E       ;BC=#5800+E
220        ADD     HL,BC     ;HL=Dir. en el A.FILE
230        LD      (HL),A    ;Pone atributos
240        POP     HL        ;Rec. dir. del carac.
250        PUSH    DE        ;Guarda coordenadas
260        LD      A,D       ;A=LINEA (Y)
270        AND     #18       ;Max. linea=24
280        ADD     A,#40     ;A=Byte alto del D.F.
290        LD      B,A
300        LD      A,D       ;A=linea
310        RRCA    ;         ;Pasa los bits
320        RRCA    :         ;  0,1 y 2 a la
330        RRCA    ;         ;  parte alta
340        AND     #E0       ;Borra el resto
350        ADD     A,E       ;Le suma la columna
360        LD      E,A       ;E=Byte bajo del D.F.
370        LD      D,B       ;D=Byte alto del D.F.
380        LD      B,8       ;Lineas del caracter
390 BUPINT LD      A,(HL)    ;A=Byte del caracter
400        LD      (DE),A    ;Lo pone en el D.FILE
410        INC     D         ;Prox. linea DIS.FILE
420        INC     HL        ;Prox. byte del car.
430        DJNZ    BUPINT    ;Repite bucle 8 veces
440        POP     DE        ;Recupera coordenadas
450 ;
460 ;          *  INCREMENTA COORDENADAS  *
470
480 INCDE  LD      A,E       ;A=Columna (X)
490        INC     A         ;La incrementa
500        AND     31        ;Si es menor de 32
510        LD      E,A
520        RET     NZ        ;  retorno
530        INC     D         ;Incrementa linea
540        LD      A,D
550        CP      24        ;Si es menor de 24
560        RET     C         ;  retorna
570        DEC     D         ;Recupera valor
580        PUSH    DE        ;  si fin pantalla
590        CALL    SCROLL    ;  scroll arriba
600        POP     DE
610        RET
620 ;
630 CHARS  EQU     23606     ;Dir. tabla caract.
640 SCROLL EQU     3582      ;Scroll arriba
```

PRINT en alta resolución

Esta rutina permitirá imprimir un caracter, en cualquier coordenada de la pantalla en alta resolución.

Se utiliza haciendo:

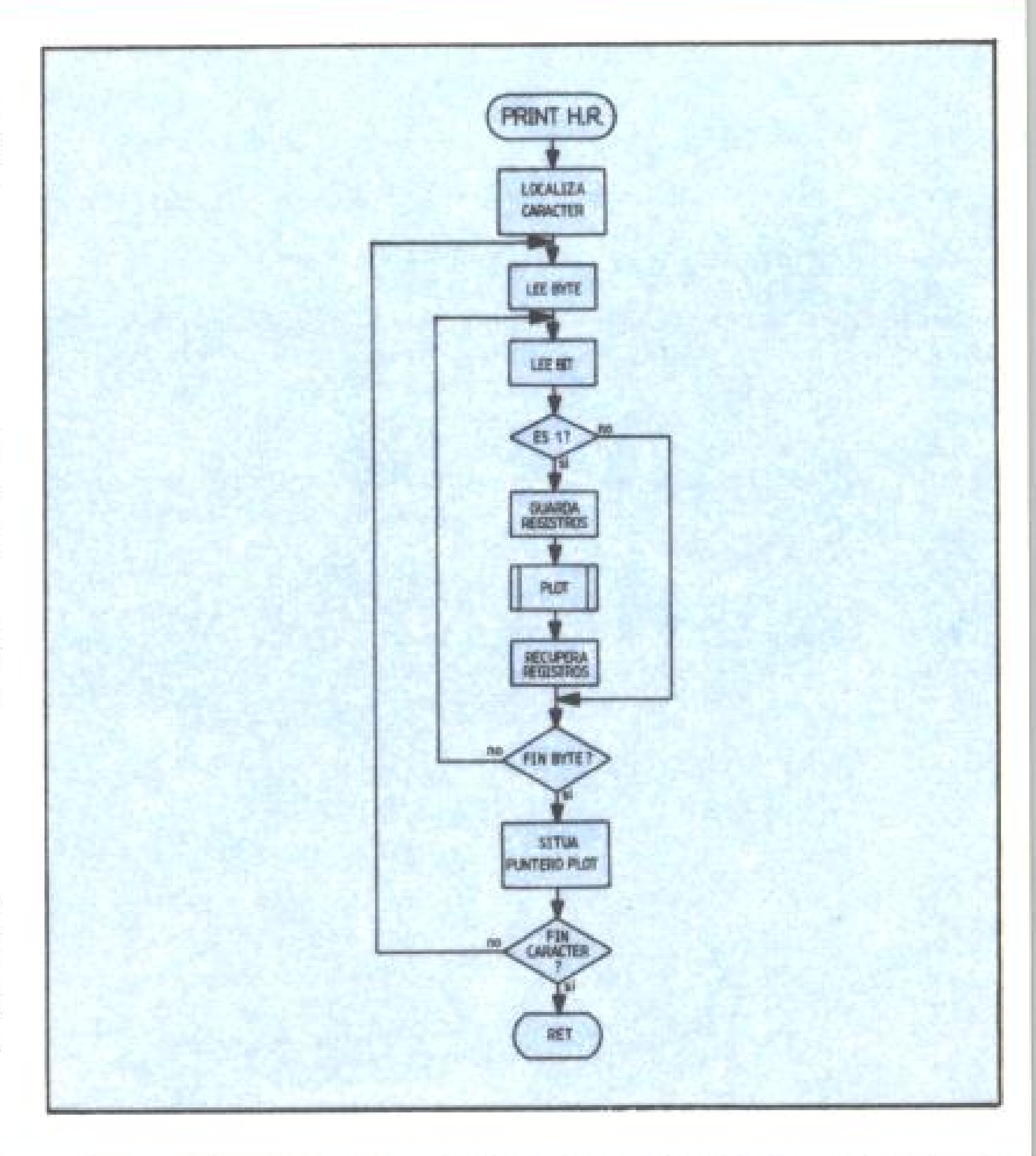
PLOT INVERSE 1; X, Y: POKE 23681,C:
RANDOMIZE USR N

X e Y son las coordenadas donde deseamos imprimir, C es el código del caracter y N la dirección donde se encuentre la rutina (es reubicable).

El caracter se sobreimprime sobre lo que haya en la pantalla en ese momento (modo OR) de forma distinta a OVER 1 (modo XOR).

Funcionamiento:

Busca la dirección de comienzo del caracter y uno a uno va comprobando los 8 bits de cada byte. Si el bit es 1 pinta un punto (PLOT) en las coordenadas correspondientes, si es 0 no lo hace.



```
10 ;** H.R. PRINT **
20 ;
30 ;
40        ORG     60000      ;RUTINA REUBICABLE
50 ;
60        LD      HL,(23681);L=Cod. del caracter
70        LD      H,0
80        LD      DE,(COORDS);E=X   D=Y
90 ;
100 START PUSH    DE         ;Guarda coordenadas
110       LD      DE,(CHARS);Comzo. caracteres
120       ADD     HL,HL      ;Multiplica HL por 8
130       ADD     HL,HL
140       ADD     HL,HL
150       ADD     HL,DE      ;HL=Comzo. del carac.
160       POP     DE         ;Recupera coordenadas
170 ;
180       LD      B,8        ;8 bytes del caracter
190 BUCBYT LD     A,(HL)     ;Byte del caracter
200       PUSH    BC         ;Guar. cont. de bytes
210       LD      B,8        ;8 bits
220 BUCBIT PUSH   BC         ;Guar. cont. bits.
230       RLA     ;          ;Desplaza un bit
240       JR      NC,NOPLOT;Si era 0 no pinta
250       LD      B,D        ;B=Y
260       LD      C,E        ;C=X
270       PUSH    DE         ;Guarda registros
280       PUSH    HL
290       PUSH    AF
300       CALL    PLOT       ;Hace PLOT C,B
310       POP     AF         ;Rec. byte del carac.
320       POP     HL         ;Rec. dir. del byte
330       POP     DE         ;Rec. coordenadas
340 NOPLOT INC    E          ;Incrementa X
350       POP     BC         ;Rec. cont. bits
360       DJNZ    BUCBIT     ;Proximo bit
370       DEC     D          ;Decrementa Y
380       POP     BC         ;Rec. cont. de bytes
390       INC     HL         ;Dir. del byte
400       LD      A,248      ;A=-8
410       ADD     A,E        ;Resta 8 a X
420       LD      E,A
430       DJNZ    BUCBYT     ;Proximo byte
440       RET     ;          ;Vuelve al BASIC
450 ;
460 ;
470 ;
480 COORDS EQU    23677      ;X e Y del ult. PLOT
490 PLOT  EQU     #22E5      ;Dibuja un punto
500 CHARS EQU     23606      ;Dir. tabla caract.
```

```
10 DATA "2A 81 5C 26 00 ED 5B 7D",754
20 DATA "5C D5 ED 5B 36 5C 29 29",861
30 DATA "29 19 D1 06 08 7E C5 06",618
40 DATA "08 C5 17 30 0B 42 4B D5",641
50 DATA "E5 F5 CD E5 22 F1 E1 D1",1617
60 DATA "1C C1 10 ED 15 C1 23 3E",785
70 DATA "F8 83 5F 10 E0 C9      ",915
```

PRINT caracter ampliado

Con esta rutina se pueden imprimir caracteres en cualquier escala de ampliación en la pantalla y en cualquier dirección de alta resolución.

Se utiliza haciendo:

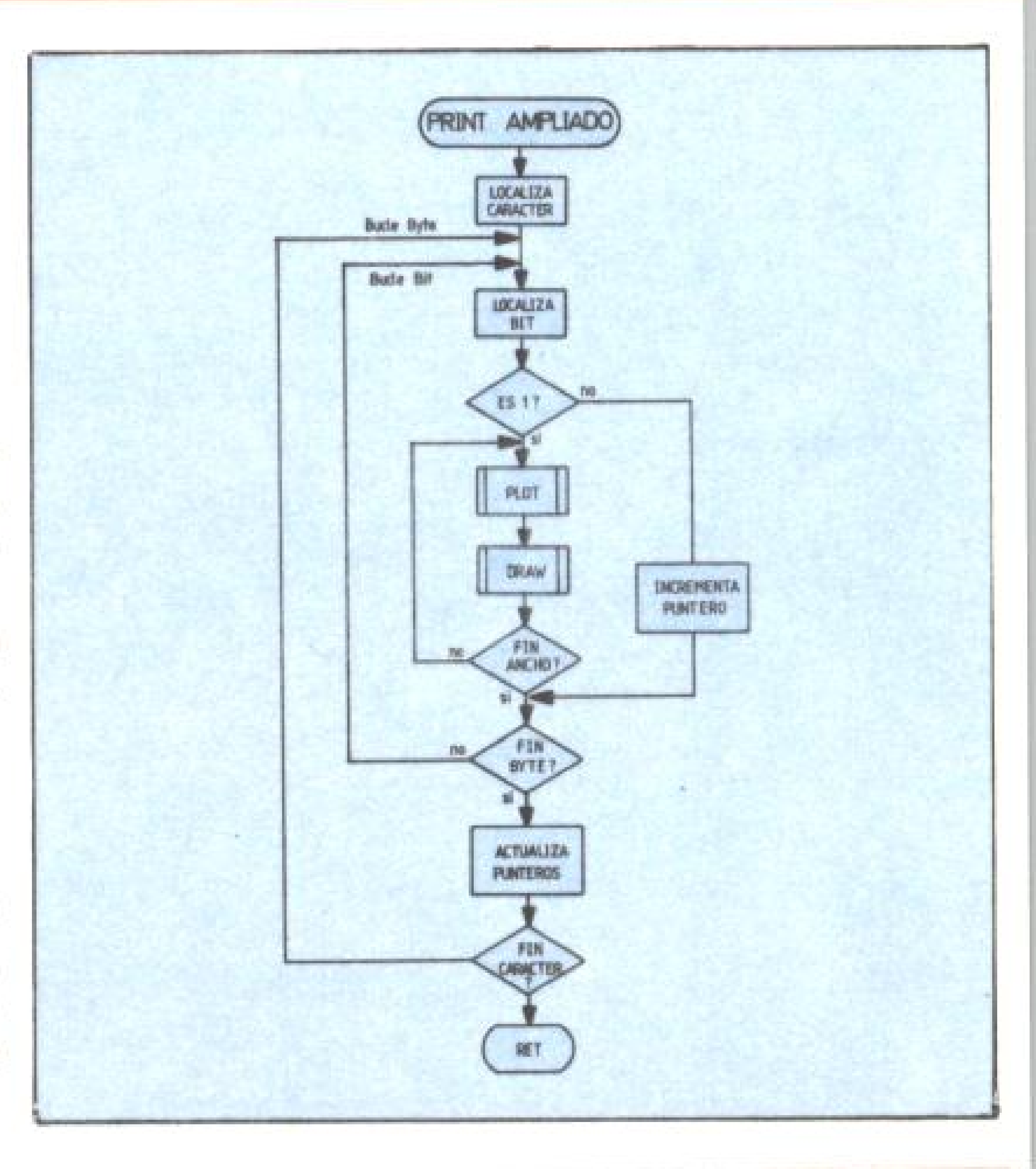
RANDOMIZE A + H * 256: PLOT INVERSE 1; X,Y: POKE 23681, C: LET B = USR N

A y H son el ancho y alto, X e Y son las coordenadas donde deseamos imprimir, C es el código del caracter y N la dirección donde se encuentre la rutina (es reubicable).

Para escribir un texto debe incrementarse a cada caracter la coordenada X en 8 veces el ancho.

Funcionamiento:

Recorre la definición del caracter comprobando cada uno de los 64 bits que lo componen. Cada vez que encuentra un 1 dibuja tantas líneas como hayamos indicado de ancho, de una longitud correspondiente al alto.



```
 10 ;** PRINT AMPLIADU **
 20         ORG     60000     ;RUTINA REUBICABLE
 30         LD      HL,(23681);L=Cod. del caracter
 40         LD      H,0
 50         LD      DE,(CHARS);Comzo. caracteres
 60 START   ADD     HL,HL     ;Multiplica HL por 8
 70         ADD     HL,HL
 80         ADD     HL,HL
 90         ADD     HL,DE     ;HL=Comzo. del carac.
100         LD      DE,(COORDS);E=X    D=Y
110         LD      B,8       ;8 bytes del caracter
120 BUCBYT  LD      A,(HL)    ;Byte del caracter
130         PUSH    BC        ;Guar. cont. de bytes
140         LD      B,8       ;8 bits
150 BUCBIT  PUSH    BC        ;Guar. cont. bits
160         RLA     ;         ;Desplaza un bit
170         JR      NC,NOPLOT;Si era 0 no pinta
180         LD      BC,(23669);B=ANCHO
190 ANCHO   PUSH    BC        ;Guarda cont. ancho
200         LD      B,D       ;B=Y
210         LD      C,E       ;C=x
220         PUSH    DE        ;Guarda registros
230         PUSH    HL
240         PUSH    AF
250         CALL    PLOT      ;Hace PLOT C,B
260         EXX     ;         ;Interc. registros
270         PUSH    HL        ;Guarda HL'
280         EXX     ;         ;Restable registros
290         LD      BC,(23670);B=ALTO
300         LD      C,0
310         LD      DE,#01FF  ;DRAW 0,-B
320         CALL    DRAW      ;Dibuja la linea
330         EXX     ;         ;Interc. registros
340         POP     HL        ;Recupera HL'
350         EXX     ;         ;Interc. registros
360         POP     AF        ;Rec. byte del carac.
370         POP     HL        ;Rec. dir. del byte
380         POP     DE        ;Rec. coordenadas
390         POP     BC        ;Rec. cont. de ancho
400         INC     E         ;Incrementa X (ancho)
410         DJNZ    ANCHO     ;Bucle del ancho
420         JR      PROXBI    ;Proximo bit
430 NOPLOT  LD      BC,(23669);B=ANCHO
440 INCAN   INC     E         ;Incrementa la X
450         DJNZ    INCAN     ;  sin dibujar lineas
460 PROXBI  POP     BC        ;Rec. cont. de bits
470         DJNZ    BUCBIT    ;Bucle de rotacion
480         LD      BC,(23670);B=ALTO
490 DECALT  DEC     D         ;Decrementa la Y
500         DJNZ    DECALT    ;Hasta situarse abajo
510         INC     HL        ;Dir. del byte
520         LD      BC,(23669);ANCHO
530         RL      B         ;Multiplica
540         RL      B         ;  el ancho
550         RL      B         ;  por 8
560 RESTAN  DEC     E         ;Restablece la
570         DJNZ    RESTAN    ;coordenada X
580         POP     BC        ;Rec. cont. de bytes
590         DJNZ    BUCBYT    ;Proximo byte
600         RET     ;         ;Vuelve al BASIC
610 CHARS   EQU     23606     ;Dir. tabla caract.
620 COORDS  EQU     23677     ;X e Y del ult. PLOT
630 PLOT    EQU     #22E5     ;Dibuja un punto
640 DRAW    EQU     #24BA     ;Dibuja una linea
```

```
 10 DATA "2A 81 5C 26 00 ED 5B 36",683
 20 DATA "5C 29 29 29 19 ED 5B 7D",693
 30 DATA "5C 06 08 7E C5 06 08 C5",640
 40 DATA "17 30 28 ED 4B 75 5C C5",829
 50 DATA "42 4B D5 E5 F5 CD E5 22",1296
 60 DATA "D9 E5 D9 ED 4B 76 5C 0E",1199
 70 DATA "00 11 FF 01 CD BA 24 D9",917
 80 DATA "E1 D9 F1 E1 D1 C1 1C 10",1354
 90 DATA "DE 18 07 ED 4B 75 5C 1C",802
100 DATA "10 FD C1 10 CA ED 4B 76",1110
110 DATA "5C 15 10 FD 23 ED 4B 75",846
120 DATA "5C CB 10 CB 10 CB 10 1D",778
130 DATA "10 FD C1 10 AE C9      ",853
```

Lectura de teclado

Esta rutina reconoce la pulsación de una tecla, aun estando pulsada también otra.

Las teclas van numeradas del 1 al 40 de izquierda a derecha y de arriba a abajo. Si queremos conocer la pulsación de una tecla haremos:

LET A = N AND USR 60000

El valor de A será 1 si está pulsada y 0 en caso contrario.

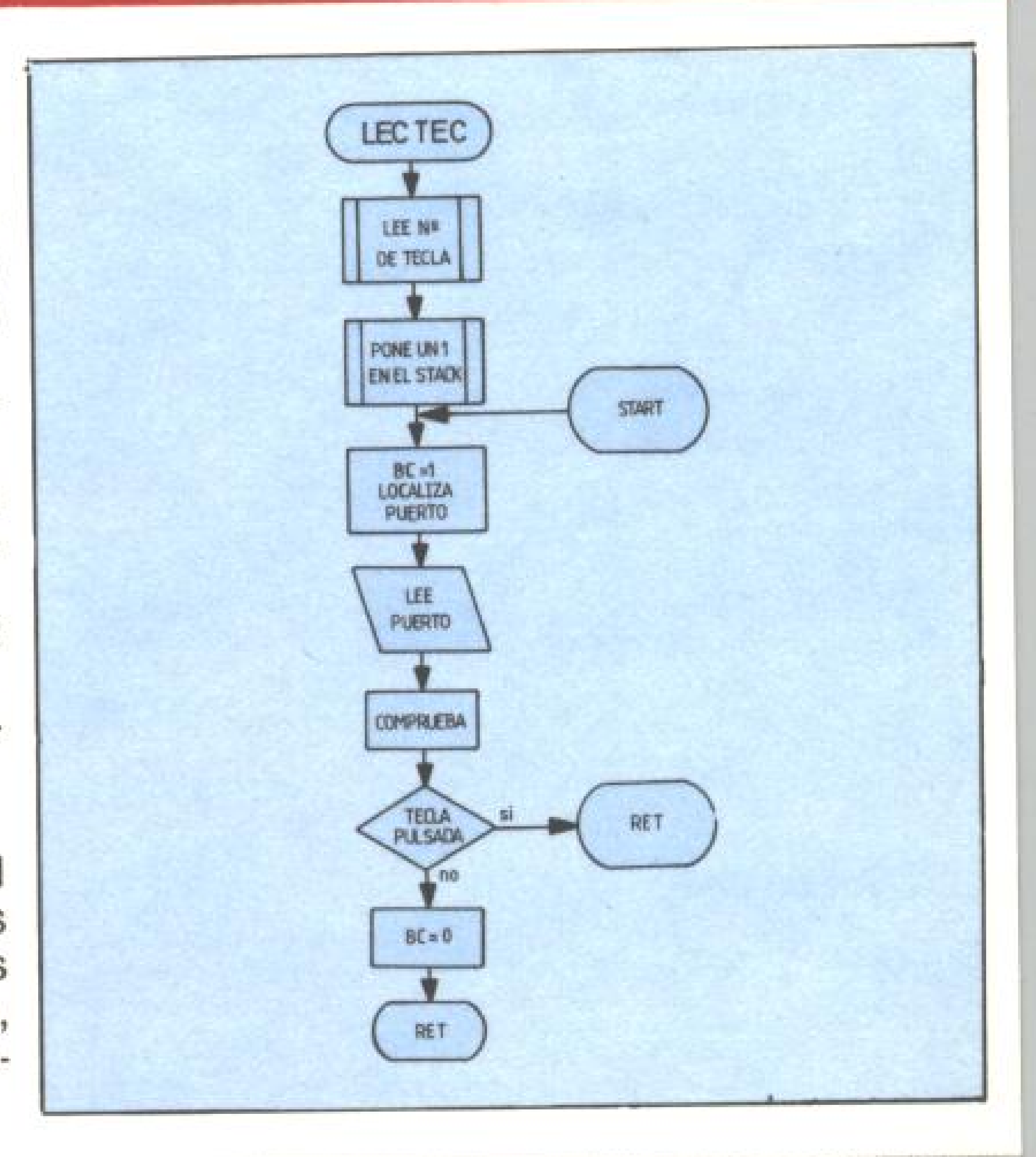
Para conocer la pulsación de varias teclas a la vez (por ejemplo el caso de mayúscula), deberemos hacer:

LET A = (N AND USR 60000) AND (M AND USR 60000)

Donde N y M son las dos teclas que queremos comprobar.

Funcionamiento

Comienza llamando a FINT 1, que lee en A el número de tecla y lo transfiere a HL, después guarda un 1 en el STACK. Busca en la tabla los datos de la tecla y comprueba si está pulsada, en este caso la función USR valdría 1, y 0 en caso contrario.



```
  1 *C-
 10 ; ** LECTURA SIMULTANEA DEL TECLADO **
 20 ;
 30 ;
 40          ORG     60000     ;RUTINA NO REUBICABLE
 50 ;
 60          CALL    FINT1     ;No. de tecla en A
 70          DEC     A         ; lo decrementa
 80          LD      H,0
 90          LD      L,A       ;HL=numero de tecla
100          PUSH    HL        ;  lo guarda
110          LD      BC,1      ;Pone un uno en
120          CALL    STKBC     ;el STK
130          POP     HL        ;Rec. num. de tecla
140 START    LD      BC,1      ;Valor del AND si
150 ;                          ;  esta pulsada
160          LD      DE,TABLA  ;DE=cmzo. tabla datos
170          ADD     HL,HL     ;Num. de tecla * 2
180          ADD     HL,DE     ;Encuentra dir. dato
190          LD      A,(HL)    ;Port de la tecla
200          INC     HL        ;Sig. dato
210          IN      A,(254)   ;Lee el teclado
220          AND     (HL)      ;Bit de la tecla
230          RET     Z         ;Ret. si estaba a 1
240          LD      BC,0      ;Si no retorna con
250          RET     ;         ;  un 0 en el AND
260 ;
270 ;
280 ;                1     2     3     4     5
290 TABLA    DEFB    247,1,247,2,247,4,247,8,247,16
300 ;                6     7     8     9     0
310          DEFB    239,16,239,8,239,4,239,2,239,1
320 ;                Q     W     E     R     T
330          DEFB    251,1,251,2,251,4,251,8,251,16
340 ;                Y     U     I     O     P
350          DEFB    223,16,223,8,223,4,223,2,223,1
360 ;                A     S     D     F     G
370          DEFB    253,1,253,2,253,4,253,8,253,16
380 ;                H     J     K     L  ENTER
390          DEFB    191,16,191,8,191,4,191,2,191,1
400 ;               C.S.   Z     X     C     V
410          DEFB    254,1,254,2,254,4,254,8,254,16
420 ;                B     N     M    S.S.  B/S
430          DEFB    127,16,127,8,127,4,127,2,127,1
440 ;
450 ;
460 ;
470 FINT1    EQU     #1E94     ;Lee num. del STK num.
480 STKBC    EQU     #2D2B     ;Guar. num. en STK
```

```
 10 DATA "CD 94 1E 3D 26 00 6F E5",822
 20 DATA "01 01 00 CD 2B 2D E1 01",521
 30 DATA "01 00 11 81 EA 29 19 7E",573
 40 DATA "23 DB FE A6 C8 01 00 00",875
 50 DATA "C9 F7 01 F7 02 F7 04 F7",1196
 60 DATA "08 F7 10 EF 10 EF 08 EF",1012
 70 DATA "04 EF 02 EF 01 FB 01 FB",988
 80 DATA "02 FB 04 FB 08 FB 10 DF",1006
 90 DATA "10 DF 08 DF 04 DF 02 DF",922
100 DATA "01 FD 01 FD 02 FD 04 FD",1020
110 DATA "08 FD 10 BF 10 BF 08 BF",874
120 DATA "04 BF 02 BF 01 FE 01 FE",898
130 DATA "02 FE 04 FE 08 FE 10 7F",919
140 DATA "10 7F 08 7F 04 7F 02 7F",538
150 DATA "01 00 00 00 00 00 00 00",1
```

INPUT numérico

Podremos hacer la entrada de un número con visualización en cualquier lugar de la pantalla evitando la producción de errores por pulsación de teclas no numéricas.

Para usarse desde BASIC se debe crear un Buffer en una variable alfanumérica de una longitud igual al máximo de cifras admisible. La forma de llamada es:

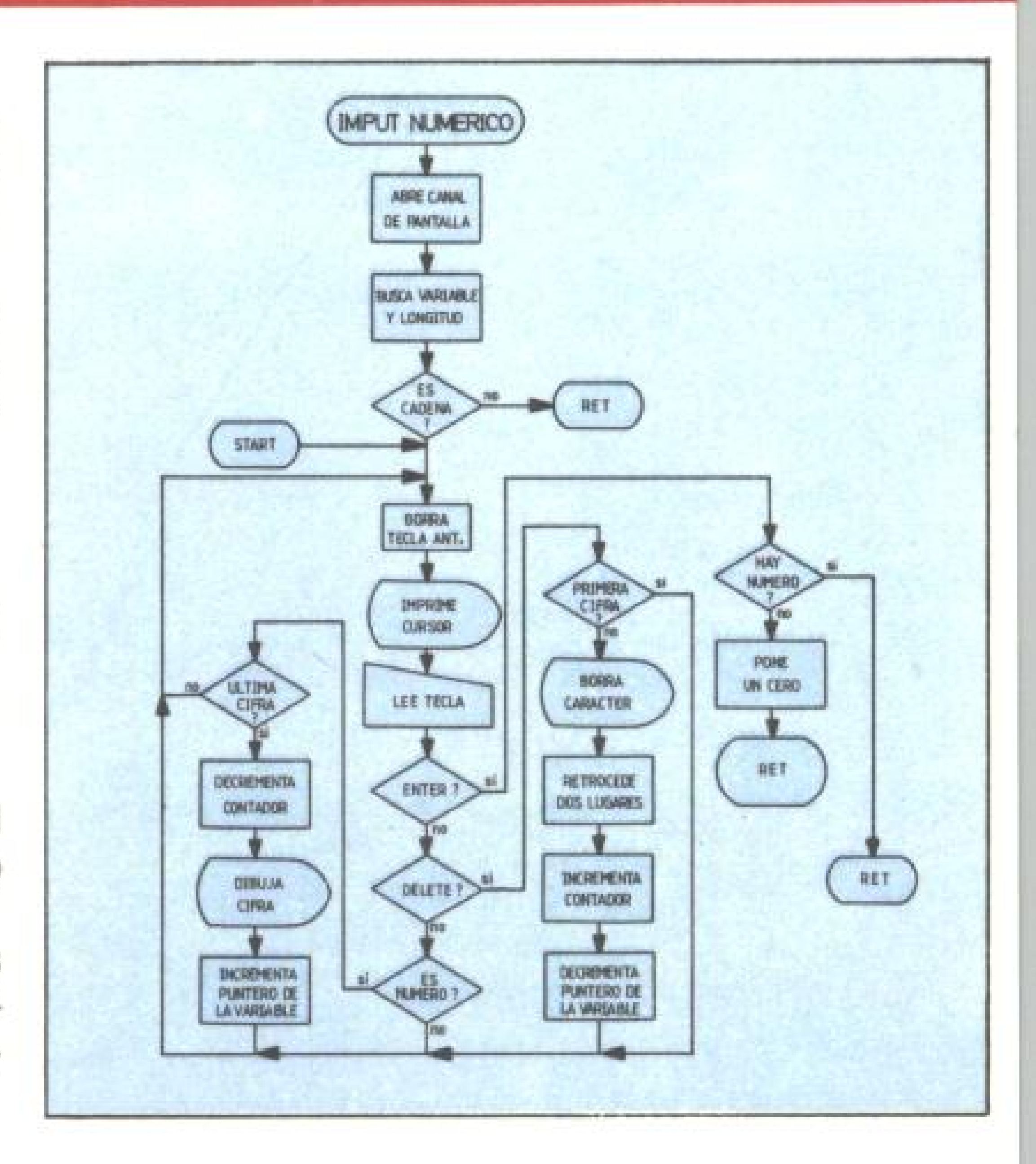
PRINT AT L,C; : LET B$ = "   ":
LET B$ = B$ AND USR N: LET I = VAL B$

L y C son las coordenadas de presentación, B$ el Buffer, N la dirección de la rutina (es reubicable), e I la variable numérica.

Funcionamiento:

En primer lugar comprueba si está creado el buffer en una variable alfanumérica retornando en caso contrario.

Posteriormente atiende solamente las teclas numéricas, Delete y Enter dibujando tras los números un cursor. Si es pulsado Enter sin número asigna el valor 0.



```
 10 ;** INPUT C/M **
 20         ORG     60000       ;RUTINA REUBICABLE
 30         LD      A,2         ;Canal #2 (pantalla)
 40         CALL    5633        ;  lo abre
 50         LD      HL,(DEST);Comzo. variable
 60         PUSH    HL          ;Lo guarda
 70         DEC     HL
 80         DEC     HL
 90         LD      B,(HL)      ;Long. de la variable
100         LD      C,B
110         DEC     HL
120         LD      A,(HL)      ;Nombre variable
130         POP     HL          ;Rec. comzo. variable
140         AND     #E0         ;Mascara tipo var.
150         CP      #40         ;Si no es una cadena
160         RET     NZ          ;Vuelve al BASIC
170 START   XOR     A
180         LD      (LAST_K),A;Borra tecla pulsada
190         LD      A,143       ;Imprime el
200         RST     #10         ;  cursor
210         LD      A,8         ;Retrocede un
220         RST     #10         ;  caracter
230         LD      A,(LAST_K);A=Cod. tecla puls.
240         CP      13          ;Si pulsa ENTER
250         JR      Z,FIN       ;  acaba el INPUT
260         CP      12          ;Cod. de DELETE
270         JR      Z,DELETE
280         CP      48          ;Si no es tecla
290         JR      C,START     ;  numerica vuelve
300         CP      58          ;  al test
310         JR      NC,START
320         DEC     B           ;Si es 0, Z=0
330         INC     B           ;Restablece B
340         JR      Z,START     ;Vuelve al test
350         DEC     B           ;Caract. que quedan
360         LD      (HL),A      ;Guar. el no. pulsado
370         RST     #10         ;  y lo imprime
380         INC     HL          ;Dir. en la variable
390         JR      START       ;Comprueba el teclado
400 DELETE  LD      A,B         ;Si esta al comienzo
410         CP      C           ;  no retrocede mas
420         JR      Z,START     ;  y vuelve a START
430         LD      A,32        ;Guarda un espacio
440         DEC     HL          ;Restablece los
450         LD      (HL),A      ;En la variable
460         RST     #10         ;Borra el caracter
470         LD      A,8         ;Retrocede dos
480         RST     #10         ;  caracteres
490         LD      A,8
500         RST     #10
510         INC     B           ;  puntero
520         JR      START
530 FIN     LD      A,B         ;Caracteres pulsados
540         CP      C           ;Si escribio algo
550         RET     NZ          ;Vuelve al BASIC
560         LD      A,48        ;Si no, pone
570         LD      (HL),A      ;  un 0 en la var.
580         RET     ;           ;  y vuelve al BASIC
590 DEST    EQU     23629       ;Dir. de var. en uso
600 LAST K  EQU     23560       ;Cod. de ult. tecla
```

```
 10 DATA "3E 02 CD 01 16 2A 4D 5C",503
 20 DATA "E5 2B 2B 46 48 2B 7E E1",851
 30 DATA "E6 E0 FE 40 C0 AF 32 08",1197
 40 DATA "5C 3E 8F D7 3E 08 D7 3A",855
 50 DATA "08 5C FE 0D 28 28 FE 0C",713
 60 DATA "28 12 FE 30 38 E7 FE 3A",959
 70 DATA "30 E3 05 04 28 DF 05 77",671
 80 DATA "D7 23 18 D9 78 B9 28 D5",1049
 90 DATA "3E 20 2B 77 D7 3E 08 D7",756
100 DATA "3E 08 D7 04 18 C7 78 B9",817
110 DATA "C0 3E 30 77 C9            ",622
```

SCROLL arriba en alta resolución

Produce un desplazamiento de la pantalla (sin atributos) hacia arriba de una línea de pixels. La rutina se puede llamar de la forma:

RANDOMIZE USR N

N es la dirección donde se encuentre la rutina (es reubicable).

Funcionamiento:

Desplaza hacia arriba una a una las 191 líneas de pixels mediante el bucle BLNPIX. BCOLUM, que está en su interior, desplaza cada línea byte borrando la última línea (c = 2).

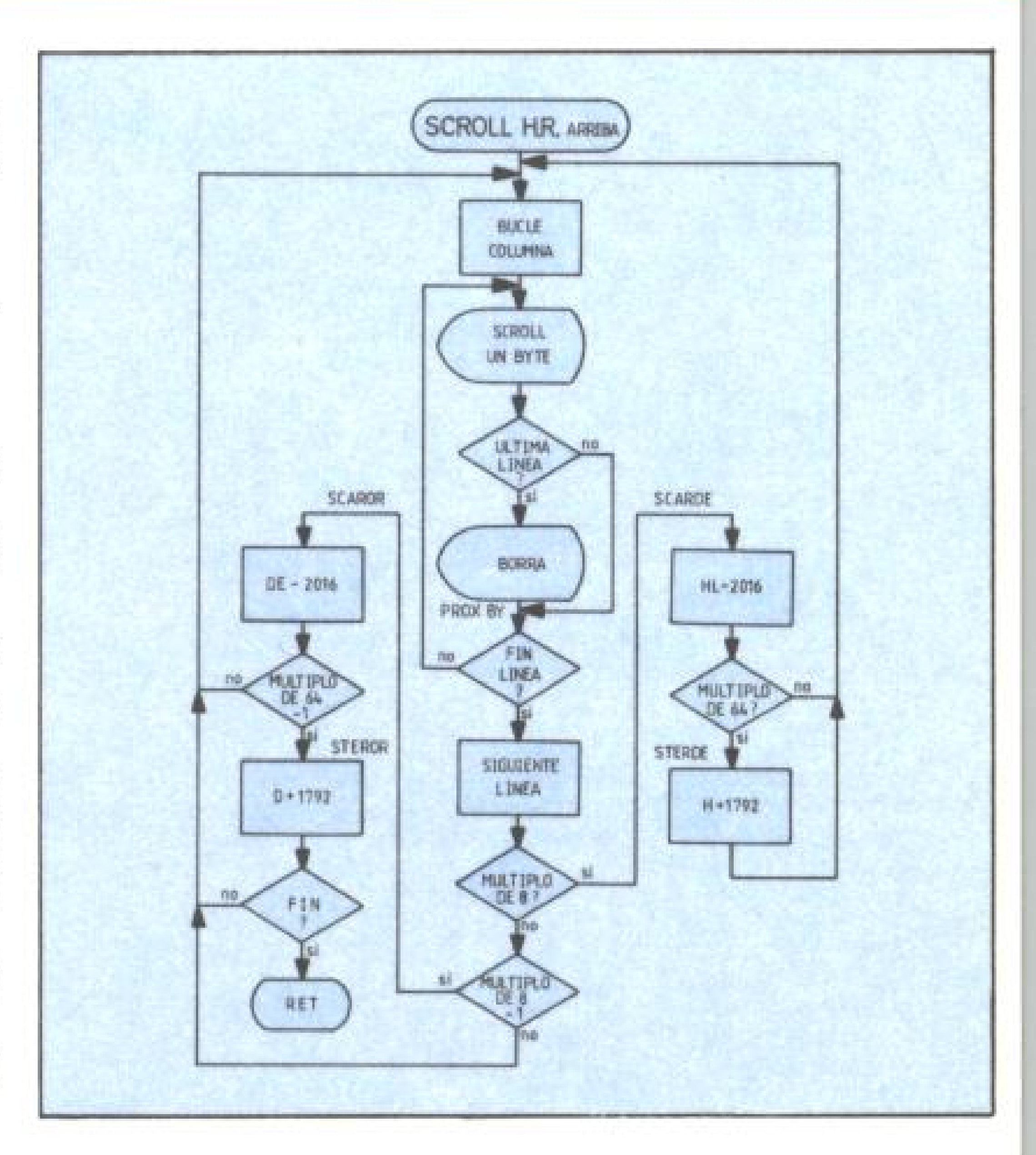
El incremento de punteros para cambiar de línea es normalmente 224 (256-32). Pero existen las siguientes excepciones:

Cuando la línea es múltiplo de 8 menos 1 cambia el caracter de origen (SCAROR): —2016.

Cuando la línea es múltiplo de 8 cambia el caracter de destino (SCARDE): —2016.

Cuando la línea es múltiplo de 64 menos 1 cambia el tercio de origen (STEROR): + 1792.

Cuando la línea es múltiplo de 64 cambia el tercio de destino (STERDE): + 1792.



```
 10 ; ** SCROLL ARRIBA EN ALTA RESOLUCION **
 20        ORG    60000
 30 START  LD     HL,16384 ;Prim. byte del DISP.F.
 40        LD     DE,16640 ;Una linea abajo
 50        LD     C,192    ;Numero de lineas
 60 BLNPIX LD     B,32     ;Contador de columnas
 70 BCOLUM LD     A,(DE)   ;Byte de origen
 80        LD     (HL),A   ;Lo POKEa en destino
 90        LD     A,C      ;Contador de lineas
100        CP     2        ;Comp. si es la ulti.
110        JR     NZ,PROXBY;Si no, prox. columna
120        XOR    A        ;Si era la ultima
130        LD     (DE),A   ;La borra
140 PROXBY INC    DE       ;Puntero de origen
150        INC    HL       ;Puntero de destino
160        DJNZ   BCOLUM   ;Una linea completa
170        PUSH   DE       ;Guar. punt. origen
180        LD     DE,224   ;Dist. a prox. linea
190        ADD    HL,DE    ;HL=Proxima linea
200        EX     (SP),HL  ;Recupera DE en HL
210        ADD    HL,DE    ;HL=Proxima linea
220        EX     DE,HL    ;Intercamb. registros
230        POP    HL       ;Rec. puntero destino
240        DEC    C        ;Contador de lineas
250        LD     A,C      ;Si la linea es
260        AND    7        ;  un multiplo de 8
270        JR     Z,SCARDE ;Sig.caracter destino
280        CP     1        ;Si es mult. de 8 -1
290        JR     Z,SCAROR ;Sig.caracter origen
300        JR     BLNPIX   ;Sig.lin. de pixels
310 SCARDE PUSH   DE       ;Guar. puntero origen
320        LD     DE,2016  ;2K-32
330        SBC    HL,DE    ;HL=Prox. lin. cars.
340        POP    DE       ;Rec. puntero origen
350        LD     A,C      ;Contador de lineas
360        AND    63       ;Si no es mult.de 64
370        JR     NZ,BLNPIX; siguiente linea
380 STERDE LD     A,7      ;Suma 792 al
390        ADD    A,H      ;  destino, para
400        LD     H,A      ;  cambiar de tercio
410        JR     BLNPIX   ;Sig.lin. de pixels
420 SCAROR PUSH   HL       ;Guar. puntero destino
430        EX     DE,HL    ;Intercamb. registros
440        LD     DE,2016  ;2K-32
450        SBC    HL,DE    ;HL=prox. lin. cars.
460        EX     DE,HL    ;Interc. registros
470        POP    HL       ;Rec. puntero origen
480        LD     A,C      ;Contador de lineas
490        AND    63       ;Si no es multiplo
500        CP     1        ;  de 64 menos 1
510        JR     NZ,BLNPIX; siguiente linea
520 STEROR LD     A,7      ;Suma 1792 al
530        ADD    A,D      ;  origen para
540        LD     D,A      ;  cambiar de tercio
550        LD     A,C      ;Contador de lineas
560        CP     1        ;Si no ha acabado
570        JR     NZ,BLNPIX; siguiente linea
580        RET
```

```
 10 DATA "21 00 40 11 00 41 0E C0",385
 20 DATA "06 20 1A 77 79 FE 02 20",592
 30 DATA "02 AF 12 13 23 10 F3 D5",721
 40 DATA "11 E0 00 19 E3 19 EB E1",978
 50 DATA "0D 79 E6 07 28 06 FE 01",672
 60 DATA "28 14 18 DC D5 11 E0 07",765
 70 DATA "ED 52 D1 79 E6 3F 20 D0",1182
 80 DATA "3E 07 84 67 18 CA E5 EB",994
 90 DATA "11 E0 07 ED 52 EB E1 79",1148
100 DATA "E6 3F FE 01 20 BA 3E 07",835
110 DATA "82 57 79 FE 01 20 B1 C9",1003
120 DATA "                       ",0
```

SCROLL abajo en alta resolución

Produce un desplazamiento de la pantalla (sin atributos) hacia abajo de una línea de pixels. La rutina se puede llamar de la forma:

RANDOMIZE USR N

N es la dirección donde se encuentre la rutina (es reubicable).

Funcionamiento:

Desplaza hacia abajo una a una las 191 líneas de pixels mediante el bucle BLNPIX. BCOLUM, que está en su interior, desplaza cada línea byte a byte borrando la línea superior (c = 2).

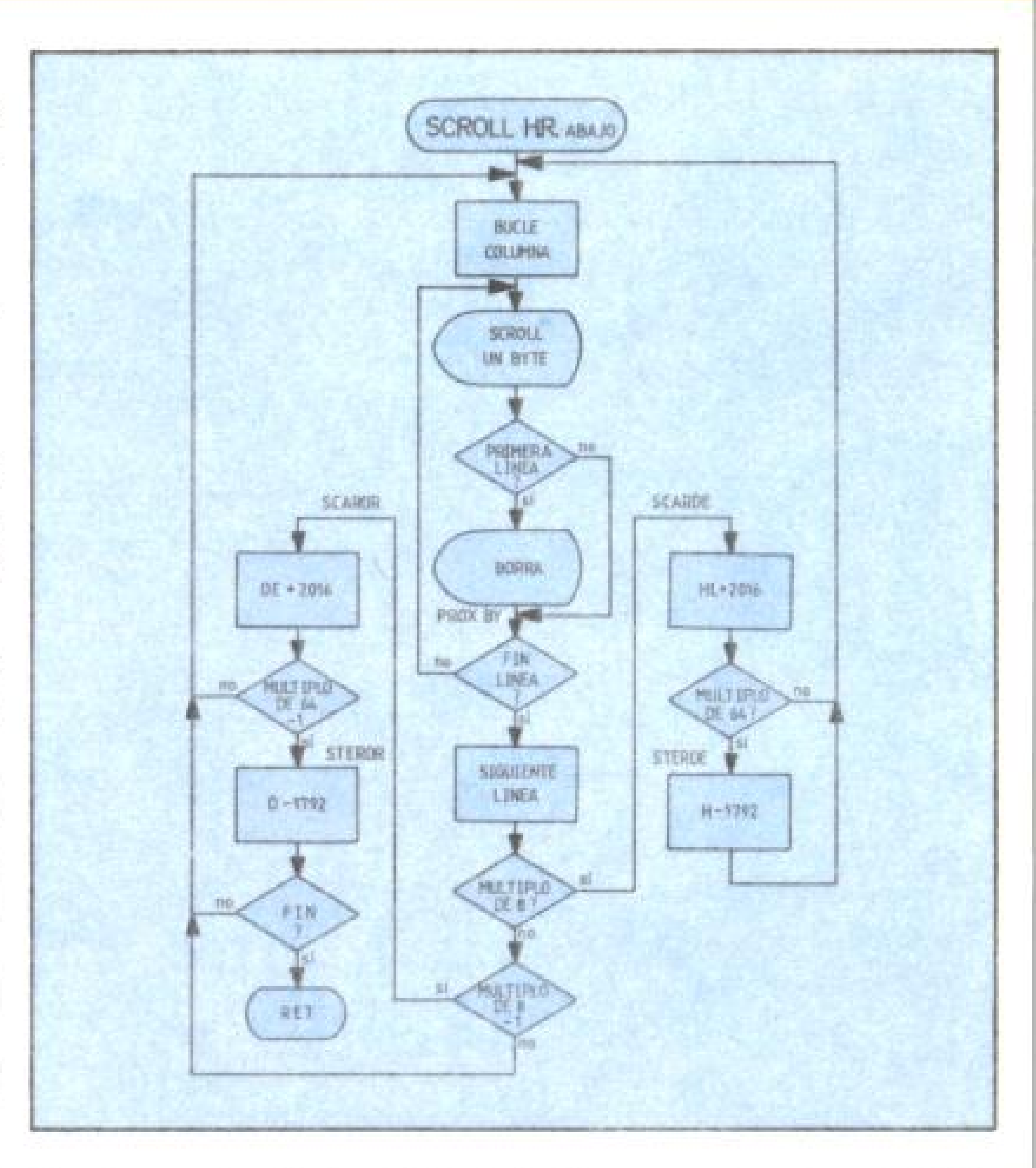
El decremento de punteros para cambiar de línea es normalmente 224 (256-32). Pero, existen las siguiente excepciones:

Cuando la línea es múltiplo de 8 menos 1 cambia el caracter de origen (SCAROR): + 2016.

Cuando la línea es múltiplo de 8 cambia el caracter de destino (SCARDE): + 2016.

Cuando la línea es múltiplo de 64 menos 1 cambia el tercio de origen (STEROR): —1792.

Cuando la línea es múltiplo de 64 cambia el tercio de destino (STERDE): —1792.



```
10 ; ** SCROLL ABAJO EN ALTA RESOLUCION **
20        ORG    60000
30 START  LD     HL,22527 ;Ult. byte del DISP.F.
40        LD     DE,22271 ;Una linea arriba
50        LD     C,192    ;Numero de lineas
60 BLNPIX LD     B,32     ;Contador de columnas
70 BCOLUM LD     A,(DE)   ;Byte de origen
80        LD     (HL),A   ;Lo POKEa en destino
90        LD     A,C      ;Contador de lineas
100       CP     2        ;Comp. si es la ulti.
110       JR     NZ,PROXBY;Si no, prox. columna
120       XOR    A        ;Si era la ultima
130       LD     (DE),A   ;La borra
140 PROXBY DEC   DE       ;Puntero de origen
150       DEC    HL       ;Puntero de destino
160       DJNZ   BCOLUM   ;Una linea completa
170       PUSH   DE       ;Guar. punt. origen
180       LD     DE,224   ;Dist. a prox. linea
190       SBC    HL,DE    ;HL=Proxima linea
200       EX     (SP),HL  ;Recupera DE en HL
210       SBC    HL,DE    ;HL=Proxima linea
220       EX     DE,HL    ;Intercamb. registros
230       POP    HL       ;Rec. puntero destino
240       DEC    C        ;Contador de lineas
250       LD     A,C      ;Si la linea es
260       AND    7        ;  un multiplo de 8
270       JR     Z,SCARDE ;Sig. carac. destino
280       CP     1        ;Si es mult. de 8 -1
290       JR     Z,SCAROR ;Sig. carac. origen
300       JR     BLNPIX   ;Sig. lin. de pixels
310 SCARDE PUSH  DE       ;Guar. puntero origen
320       LD     DE,2016  ;2K-32
330       ADD    HL,DE    ;HL=Prox. lin. cars.
340       POP    DE       ;Rec. puntero origen
350       LD     A,C      ;Contador de lineas
360       AND    63       ;Si no es mult.de 64
370       JR     NZ,BLNPIX; siguiente linea
380 STERDE LD    A,H      ;Resta 1792 al
390       SBC    A,7      ;  destino, para
400       LD     H,A      ;  cambiar de tercio
410       JR     BLNPIX   ;Siguiente linea
420 SCAROR PUSH  HL       ;Guar. puntero destino
440       LD     HL,2016  ;2K-32
450       ADD    HL,DE    ;HL=prox. lin. cars.
460       EX     DE,HL    ;Interc. registros
470       POP    HL       ;Rec. puntero origen
480       LD     A,C      ;Contador de lineas
490       AND    63       ;Si no es multiplo
500       CP     1        ;  de 64 menos 1
510       JR     NZ,BLNPIX; siguiente linea
520 STEROR LD    A,D      ;Resta 1792 al
530       SBC    A,7      ;  origen para
540       LD     D,A      ;  cambiar de tercio
550       LD     A,C      ;Contador de lineas
560       CP     1        ;Si no ha acabado
570       JR     NZ,BLNPIX; siguiente linea
580       RET
```

```
10 DATA "21 FF 57 11 FF 56 0E C0",939
20 DATA "06 20 1A 77 79 FE 02 20",592
30 DATA "02 AF 12 1B 2B 10 F3 D5",737
40 DATA "11 E0 00 ED 52 E3 ED 52",1106
50 DATA "EB E1 0D 79 E6 07 28 06",877
60 DATA "FE 01 28 13 18 DA D5 11",786
70 DATA "E0 07 19 D1 79 E6 3F 20",911
80 DATA "CF 7C DE 07 67 18 C9 E5",1117
90 DATA "21 E0 07 19 EB E1 79 E6",1100
100 DATA "3F FE 01 20 BB 7A DE 07",888
110 DATA "57 79 FE 01 20         ",495
```

SCROLL horizontal en alta resolución

Produce un desplazamiento de la pantalla (sin atributos) hacia la izquierda o derecha de un pixel. Las rutinas se pueden llamar de la forma:

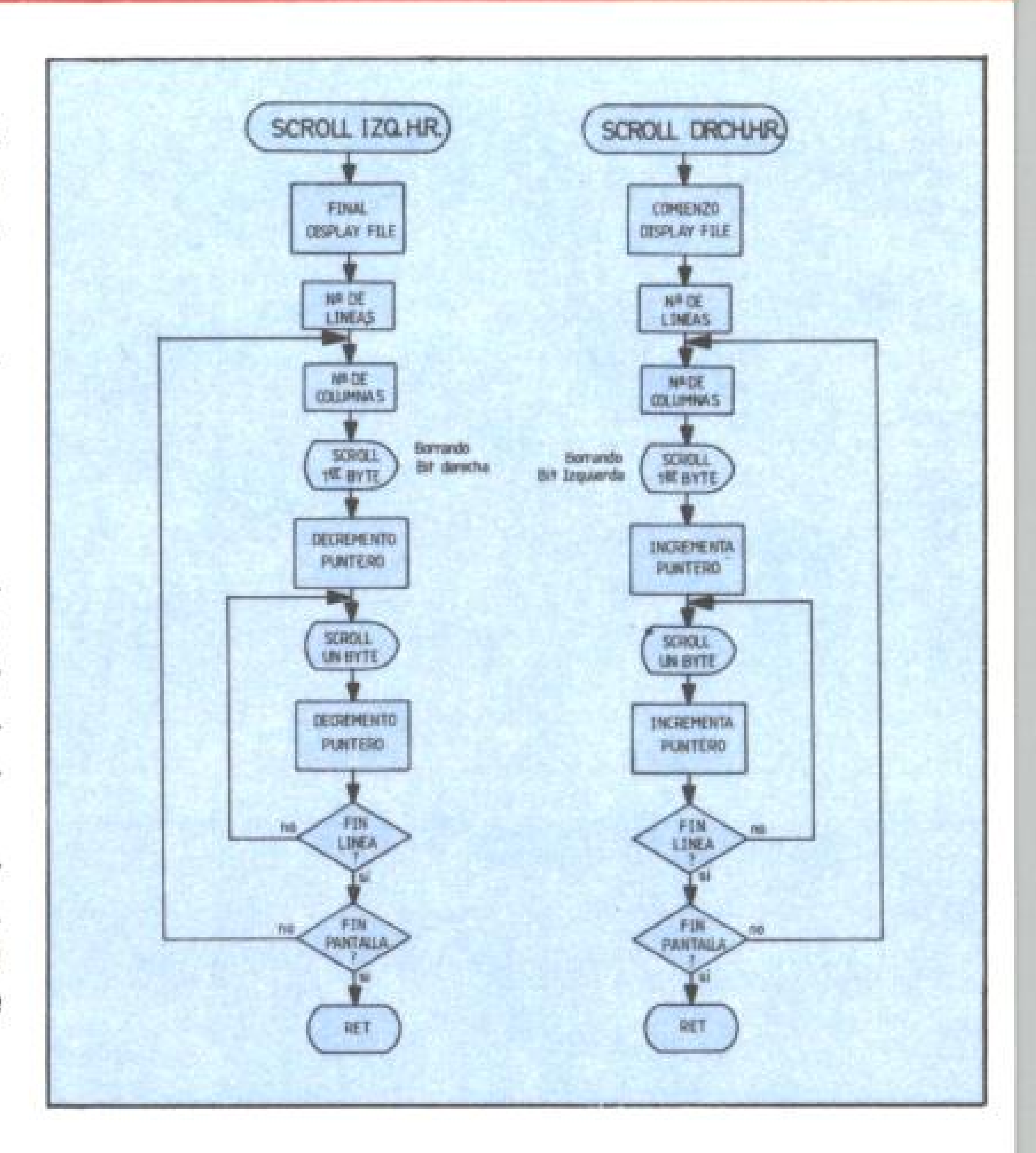
RANDOMIZE USR N

N es la dirección donde se encuentre la rutina (es reubicable).

Funcionamiento:

Desplaza hacia la izquierda o derecha un pixel de las 192 líneas de la pantalla rotando con 0 la primera vez (para borrar el bit sobrante) y con carry las 31 restantes. El scroll derecha comienza al principio de la pantalla y el de la izquierda al final.

El barrido no se hace en el orden de presentación visual sino en el del archivo de imagen. Debido a ello, si sólo se desea hacer un scroll de una parte de la pantalla deberá hacerse de un tercio completo.



```
 10 ;** SCROLL IZQUIERDA EN ALTA RESOLUCION **
 20 ;
 30         ORG     60000    ;RUTINA REUBICABLE
 40 ;
 50 START   LD      HL,22527 ;Final DISPLAY FILE
 60 ;
 70         LD      C,64*3   ;3 tercios con 64
 80 ;                        ;  lineas cada uno
 90 SHICOL  LD      B,31     ;31 columnas
100 ;
110         SLA     (HL)     ;Desp. a la izquierda
120 ;                        ;  la primera columna
130         DEC     HL       ;Puntero DISP. FILE
140 SHILIN  RL      (HL)     ;Desp. a la izquierda
150 ;
160         DEC     HL       ;Sig. columna
170         DJNZ    SHILIN   ;Scroll de linea
180 ;
190         DEC     C        ;Contador de lineas
200         JR      NZ,SHICOL;Sig. linea
210         RET
```

```
  1 *C-
 10 ;** SCROLL DERECHA EN ALTA RESOLUCION **
 20 ;
 30         ORG     60000    ;RUTINA REUBICABLE
 40 ;
 50 START   LD      HL,16384 ;Comzo. DISPLAY FILE
 60 ;
 70         LD      C,64*3   ;3 tercios con 64
 80 ;                        ;  lineas cada uno
 90 SHDCOL  LD      B,31     ;31 columnas
100 ;
110         SRL     (HL)     ;Desp. a la derecha
120 ;                        ;  la primera columna
130         INC     HL       ;Puntero DISP. FILE
140 SHDLIN  RR      (HL)     ;Desp. a la derecha
150 ;
160         INC     HL       ;Sig. columna
170         DJNZ    SHDLIN   ;Scroll de linea
180 ;
190         DEC     C        ;Contador de lineas
200         JR      NZ,SHDCOL;Sig. linea
210         RET
```

```
10 DATA "21 FF 57 0E C0 06 1F CB",821
20 DATA "26 2B CB 16 2B 10 FB 0D",629
30 DATA "20 F3 C9                ",476
```

```
10 DATA "21 00 40 0E C0 06 1F CB",543
20 DATA "3E 23 CB 1E 23 10 FB 0D",645
30 DATA "20 F3 C9                ",476
```

Utilizando esta rutina podremos tanto almacenar como volcar en pantalla cuantas figuras deseemos.

Se entiende por figura cualquier rectángulo de la pantalla sin color.

El byte MODO (60064) debe «pokearse» con 119 (carga "LD(HL),A") para archivar figuras.

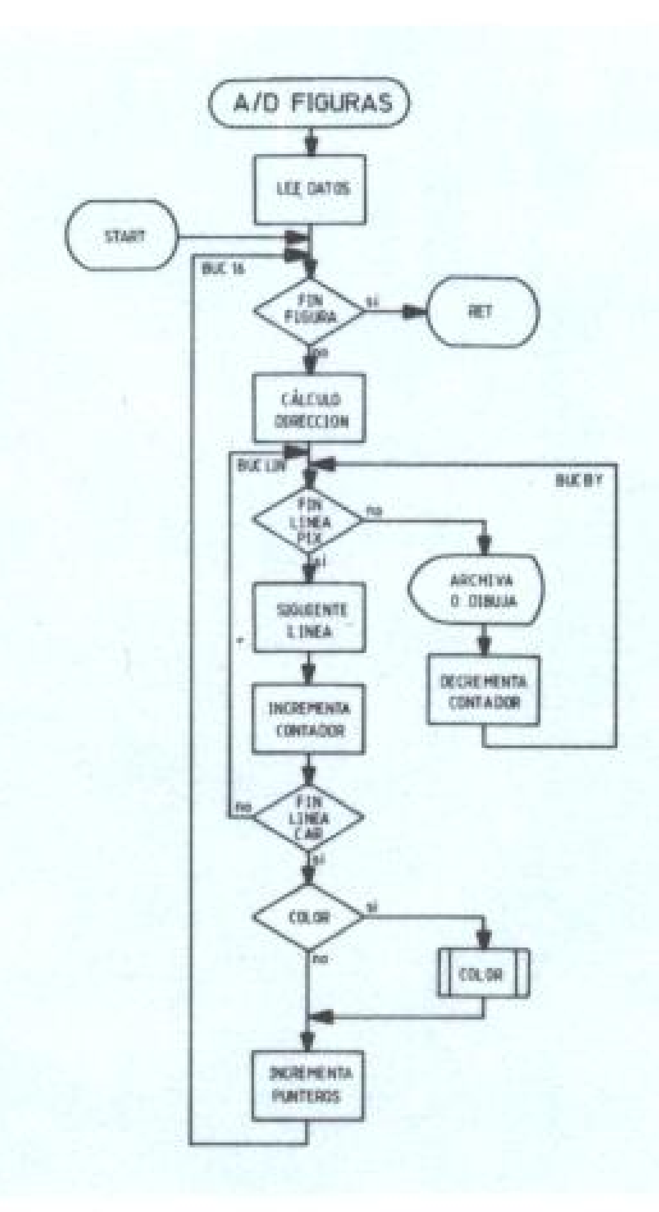
Para dibujar puede «pokearse» con 126 (copia "LD A,(HL)"), o 174 (OVER 1 "XOR (HL)"), o 182 (unión "OR (HL)"), o 166 (intersección "AND (HL)").

Para usar la rutina debe hacerse:

POKE 60001,ancho:POKE 60002, alto: RANDOMIZE dirección de archivo: PRINT AT lin,col; : RANDOMIZE USR 60000.

Funcionamiento:

Consta de tres bucles anidados. El interior (BUCBY) dibuja o archiva una línea de pixels, el siguiente (BUCLIN) una línea de caracteres, y el mayor (BUCFIG) la figura completa, calculando la dirección de cada línea de caracteres. La rutina color no se ejecuta, (ver microficha R-25).



```
 10 ;ARCHIVA/DIBUJA FIGURA
 20         ORG    60000
 30         LD     BC,#0202 ;Dimensiones
 40         LD     (TAMA),BC;Las guarda
 50         XOR    A        ;Carry flag a 0
 60         LD     HL,#1821 ;Lin 24, col 33
 70         LD     DE,(23688);P_POSN 33-c,24-1
 80         SBC    HL,DE    ;Calcula lin y col
 90         EX     DE,HL    ;Las pasa a DE
100         LD     HL,(23670);SEED (Randomize)
110         LD     A,C      ;Ancho
120         LD     (ANCHO),A;Lo guarda
130         LD     A,B      ;Alto
140         ADD    A,D      ;Lo suma a la columna
150 ;HL DIRECCION FIGURA
160 ;DE LIN/COL; BC TAMANO
170 ; A LINEA INFERIOR
180 ;
190 START  PUSH   AF       ;Guarda linea inferior
200 BUCFIG POP    AF       ;Recupera linea inferior
210        DEC    A        ;La decrementa
220        CP     D        ;Linea de pantalla
230        RET    C        ;Retorna si la pasa
240        INC    A        ;Recupera linea inferior
250        PUSH   AF       ;La guarda
260        PUSH   DE       ;Guarda linea y columna
270        LD     A,D
280        AND    #18      ;-----------------------
290        ADD    A,#40
300        LD     B,A
310        LD     A,D
320        RRCA   ;        ;Convierte linea y
330        RRCA   ;        ;columna en
340        RRCA   ;        ;direcccion en el
350        AND    #E0      ;Display file
360        ADD    A,E
370        LD     E,A
380        LD     D,B      ;-----------------------
390        LD     B,8      ;8 lineas de pixels
400 BUCLIN LD     A,(ANCHO);Ancho visible
410        LD     C,A      ;Lo carga en C
420        PUSH   DE       ;Guarda direcc. pant.
430        PUSH   HL       ;Guarda direccion fig.
440 BUCBY  LD     A,C      ;Bytes de ancho
450        CP     0        ;Linea terminada?
460        JR     Z,SIGLIN ;Siguiente linea
470        LD     A,(DE)   ;Byte de pantalla
480 MODO   XOR    (HL)     ;Diferente segun modo
490        LD     (DE),A   ;Dibuja byte
500        INC    DE       ;Inc.puntero pantalla
510        INC    HL       ;Inc.puntero figura
520        DEC    C        ;Contador ancho
530        JR     BUCBY    ;Bucle linea bytes
540 SIGLIN POP    HL       ;Recupera punt. fig.
550        LD     DE,(TAMA);Recupera ancho fig.
560        LD     D,0      ;Elimina alto
570        ADD    HL,DE    ;Sig.linea pixels
580        POP    DE       ;Recupera punt. pant.
590        INC    D        ;Siguiente lin. pixels
600        DJNZ   BUCLIN   ;Bucle lin. pixels
610        POP    DE       ;Linea y columna
620 FCOLOR OR     A        ;Carry flag a 0
630        CALL   C,XCOLOR ;Colorea lin. caract.
640        INC    D        ;Linea siguiente
650        JR     BUCFIG   ;Siguiente lin. caract.
660 TAMA   DEFS   2        ;Tamano figura
670 ANCHO  DEFS   1        ;Ancho visible
680 COLOR  DEFB   2        ;
690 XCOLOR RET    ;Ver microficha R-25
```

```
 10 DATA "01 02 02 ED 43 BB EA AF",905
 20 DATA "21 21 18 ED 5B 88 5C ED",883
 30 DATA "52 EB 2A 76 5C 79 32 BD",929
 40 DATA "EA 78 82 F5 F1 3D BA D8",1433
 50 DATA "3C F5 D5 7A E6 18 C6 40",1156
 60 DATA "47 7A 0F 0F 0F E6 E0 83",823
 70 DATA "5F 50 06 08 3A BD EA 4F",749
 80 DATA "D5 E5 79 FE 00 28 08 1A",891
 90 DATA "AE 12 13 23 0D 18 F3 E1",751
100 DATA "ED 5B BB EA 16 00 19 D1",1005
110 DATA "14 10 E1 D1 B7 DC BF EA",1298
120 DATA "14 18 C1                ",237
```

Archivo y dibujo de figuras (color)

Esta rutina debe utilizarse conjuntamente con la de archivo y dibujo de figuras (R-24).

Para que funcione debe colocarse inmediatamente detrás de ésta y activarse cambiando la instrucción OR A de la línea 620 por SCF. (POKE 60084,55).

Para desactivarse POKE 60084,183.

La rutina puede actuar de dos formas:

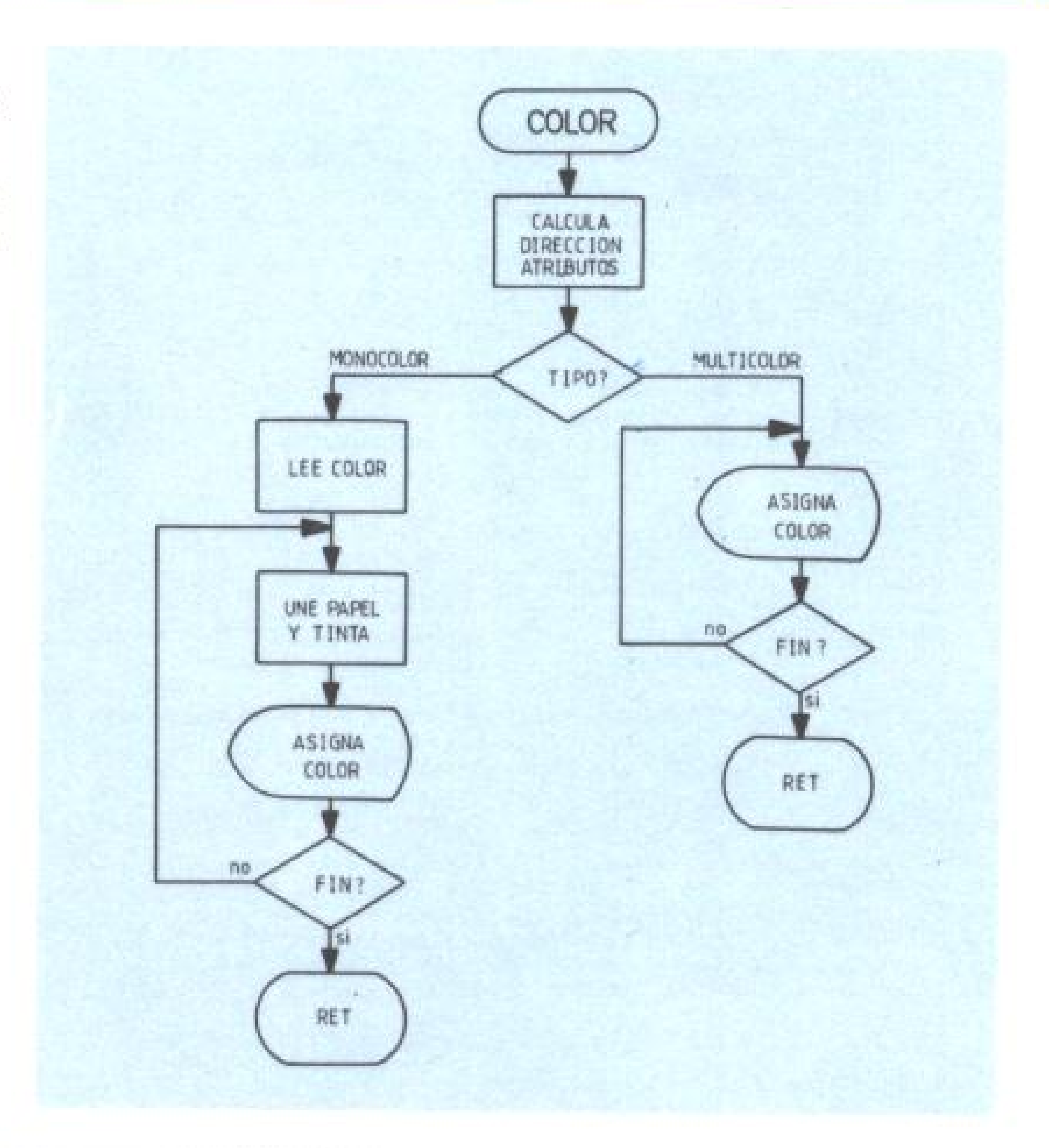
a) Color único de tinta y papel transparente:
POKE 60123,183: POKE 60094,color

b) Color múltiple (el que tenía en pantalla):
POKE 60123,55

La forma de llamada y el modo de pintado son los mismos que los de la figura sin color (microficha R-24) a lo que se deberá añadir el modo de pintado o archivo de color «pokeando» en la dirección 60144 (MODOC).

Funcionamiento:

Calcula la dirección en el fichero de atributos y entra en una de las dos rutinas para dar color a una línea de caracteres.



```
10 ;  PINTA  COLOR
20 ;
30          ORG     60093      ;Detras de a/d figura
40 ;
50 ANCHO    DEFS    1          ;Ancho visible
60 COLOR    DEFB    2          ;Codigo color
70 XCOLOR   PUSH    DE         ;Guarda lin. y col.
80          LD      A,D        ;----------------------
90          LD      D,0
100         SLA     A          ;Convierte linea y
110         SLA     A          ; columna en
120         SLA     A          ; direccion en el
130         SLA     A          ; fichero de atributos
140         RL      D
150         SLA     A
160         RL      D
170         ADD     A,E
180         LD      E,A
190         LD      A,#58
200         ADD     A,D
210         LD      D,A        ;----------------------
220         LD      A,(ANCHO);Carga ancho visible
230         LD      B,A        ;Lo pasa a B
240 ;
250 TIPO    SCF     ;          ;Carry flag a 1
260         JR      C,MULTIC ;Salta a multicolor
270 ;
280 ; DIBUJO MONOCOLOR         ;Si hay OR A en lugar
290 ;                          ; de SCF :
300         PUSH    HL         ;Guarda direcc. fig.
310         LD      A,(COLOR);Carga color
320         EX      DE,HL      ;Intercambia punteros
330         LD      D,A        ;Color
340 MONOC   LD      A,248      ;Mascara 11111000b
350         AND     (HL)       ;Atributos menos tinta
360         OR      D          ;Anade tinta
370         LD      (HL),A     ;Asigna nuevo atributo
380         INC     HL         ;inc. puntero pant.
390         DJNZ    MONOC      ;Bucle monocolor
400         POP     HL         ;Recup. punt. figura
410         POP     DE         ;Recup. punt. pantalla
420         RET     ;          ;Retorna dibujo figura
430 ;
```

```
440 ; DIBUJO MULTICOLOR
450 ;
460 MULTIC LD       A,(DE)     ;Carga color pantalla
470 MODOC  LD       A,(HL)     ;Diferente segun modo
480        LD       (DE),A     ;Asigna color pantalla
490        INC      DE         ;Inc. puntero pantalla
500        INC      HL         ;Inc. puntero figura
510        DJNZ     MULTIC     ;Bucle multicolor
520        POP      DE         ;Recup. punt. pantalla
530        RET      ;          ;Retorna dibujo figura
```

```
10 DATA "00 02 D5 7A 16 00 CB 27",601
20 DATA "CB 27 CB 27 CB 27 CB 12",947
30 DATA "CB 27 CB 12 83 5F 3E 58",839
40 DATA "82 57 3A BD EA 47 37 38",880
50 DATA "11 E5 3A BE EA EB 57 3E",1112
60 DATA "F8 A6 B2 77 23 10 F8 E1",1235
70 DATA "D1 C9 1A 7E 12 13 23 10",650
80 DATA "F9 D1 C9                ",659
```

Recorte de figuras

Colocando esta rutina inmediatamente antes de la de archivo y dibujo de figuras (microfichas R-24 y R-25) puede conseguirse hacer entradas y salidas por los laterales de la pantalla sin peligro de que se «caiga» el sistema.

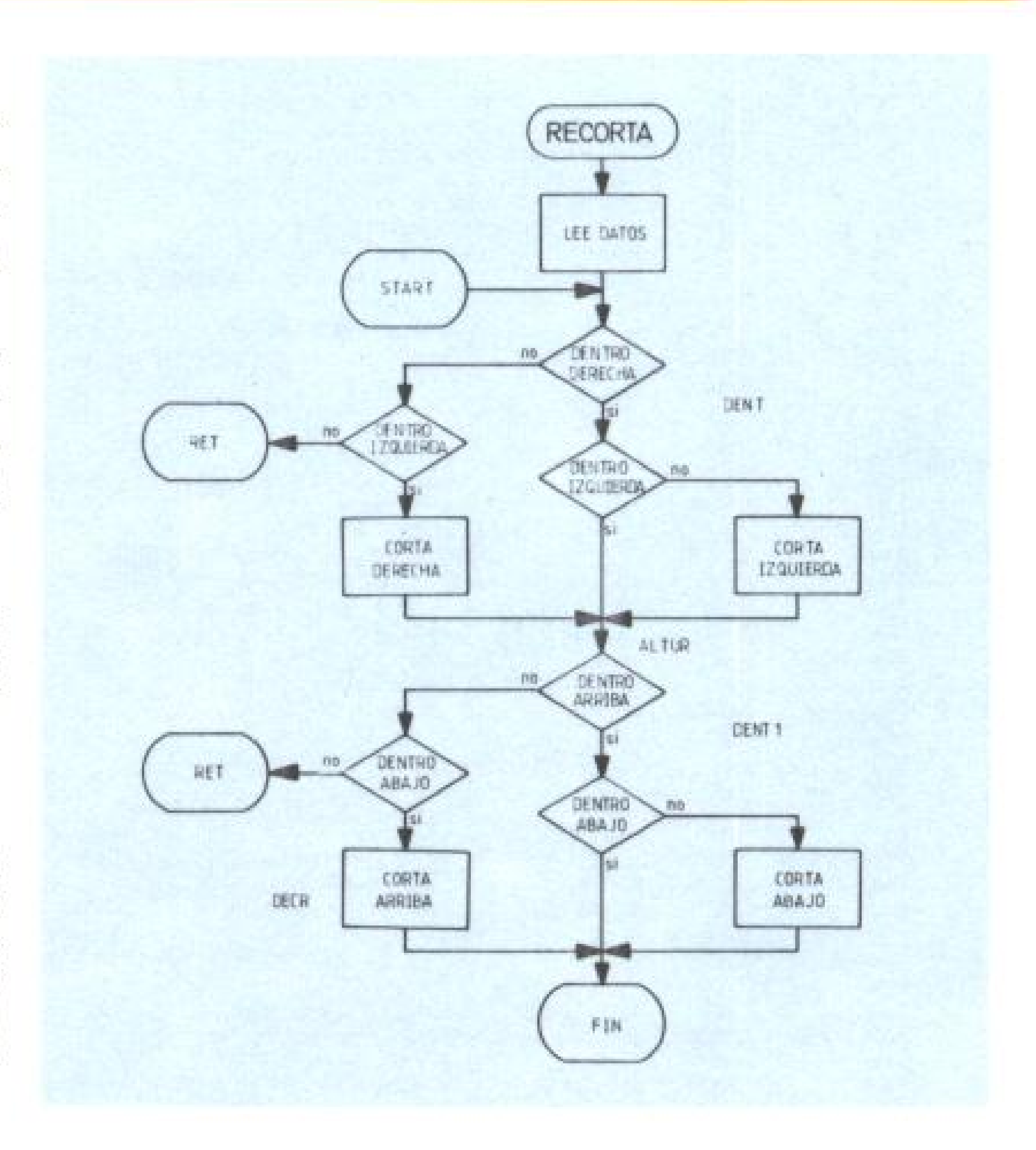
Para ensamblarlos desde Basic deben cargarse primero las rutinas de archivo de dibujo y color y, en último lugar, ésta. Posteriormente pueden salvarse conjuntamente mediante: SAVE nombre CODE 59927,225.

Para usar la rutina conjunta debe hacerse:

POKE 60001,ancho:POKE 60002,alto: POKE 23728,columna: POKE 23729,línea: RANDOMIZE dirección de archivo: RANDOMIZE USR 59927.

Funcionamiento:

En primer lugar comprueba si la figura entra dentro de la pantalla en sentido horizontal, y después en vertical. La variable ANCHO y los punteros HL (comienzo figura) y A (línea inferior) son modificados para recortar la figura. Si no puede dibujarse retorna con el flag de carry.



```
 10 ; RECORTA FIGURA
 20 ;
 30          ORG    59927      ;Delante de a/p figura
 40          LD     BC,#0606   ;Dimensiones
 50          LD     (TAMA),BC;Las guarda
 60          LD     DE,(23728);Var. del sist. no usada
 70          LD     HL,(23670);SEED
 80 START    LD     A,E        ;Columna
 90          CP     32         ;Limite derecho
100          JR     C,DENT     ;La derecha esta dentro
110          ADD    A,C        ;Suma ancho
120          RET    Z          ;Fuera de pantalla
130          CCF
140          RET    C          ;Fuera de pantalla
150          LD     (ANCHO),A;Solo parte derecha
160          LD     E,0        ;Columna 0
170          NEG    ;          ;Complementa A
180          ADD    A,C        ;  A=C-A
190          PUSH   DE         ;Guarda punt. pantalla
200          LD     D,0        ;Elimina D
210          LD     E,A        ;Bytes fuera pant.
220          ADD    HL,DE      ;Inc. punt. figura
230          POP    DE         ;Recupera punt. pant.
240          JR     ALTUR      ;Comprobacion de altura
250 DENT     LD     A,C        ;Ancho
260          LD     (ANCHO),A;Lo guarda
270          ADD    A,E        ;Lo suma a la col.
280          CP     32         ;Limite derecho
290          JR     C,ALTUR    ;Salta si no lo supera
300          LD     A,32       ;Columna 32
310          SUB    E          ;La resta a la col actual
320          LD     (ANCHO),A;Ancho visible
330 ALTUR    LD     A,D        ;Numero de linea
340          CP     24         ;Linea inferior
350          JR     C,DENT1    ;Parte sup. dentro
360          ADD    A,B        ;Suma alto
370          RET    Z          ;Fuera de pantalla
380          CCF
390          RET    C          ;Fuera de pantalla
400          PUSH   AF         ;Guarda abajo
410          LD     A,0        ;( D es negativo )
420          SUB    D          ;A = ABS (D)
430          LD     D,0        ;Parte sup. de pant.
440          PUSH   BC         ;Guarda dimensiones
450          LD     B,0
460          LD     C,A        ;BC =alto sobrante
470          LD     A,(TAMA)   ;Ancho figura real
480          SLA    A          ; A*2
490          SLA    A          ; A*4
500          SLA    A          ; A*8
510 DECR     ADD    HL,BC      ;--------------------
520          DEC    A          ;Corta parte superior
530          JR     NZ,DECR    ;--------------------
540          POP    BC         ;Recupera dimensiones
550          JR     FIN
560 DENT1    LD     A,B        ;Altura
570          ADD    A,D        ;Parte inferior
580          CP     24         ;Esta dentro?
590          JR     C,FIN2     ;Si esta dentro
600          LD     A,24       ;Linea inferior
610 FIN2     PUSH   AF         ;Guarda linea inf.
620 FIN      POP    AF         ;Recupera lin. inf.
630          OR     A          ;Carry a 0
640 ;
650 TAMA     EQU    60091
660 ANCHO    EQU    60093
```

```
 10 DATA "01 06 06 ED 43 BB EA ED",975
 20 DATA "5B B0 5C 2A 76 5C 7B FE",988
 30 DATA "20 38 14 81 C8 3F D8 32",766
 40 DATA "BD EA 1E 00 ED 44 81 D5",1100
 50 DATA "16 00 5F 19 D1 18 0F 79",511
 60 DATA "32 BD EA 83 FE 20 38 06",952
 70 DATA "3E 20 93 32 BD EA 7A FE",1090
 80 DATA "18 38 1E 80 C8 3F D8 F5",962
 90 DATA "3E 00 92 16 00 C5 06 00",433
100 DATA "4F 3A BB EA CB 27 CB 27",1042
110 DATA "CB 27 09 3D 20 FC C1 18",813
120 DATA "09 78 82 FE 18 38 02 3E",657
130 DATA "18 F5 F1 B7             ",693
```

Con la ayuda de esta rutina podremos simular tanto en código máquina como en BASIC desplazamientos rectilíneos de móviles de una forma similar a como lo hace la rutina DRAW.

● En código máquina puede hacerse una tabla con varios móviles indizada con IX:

IX + 0 Código que utiliza la rutina y debe inicializarse con 255 siendo respetado las siguientes veces que sea llamada.

IX + 1 Coordenada X actual.

IX + 2 Coordenada Y actual.

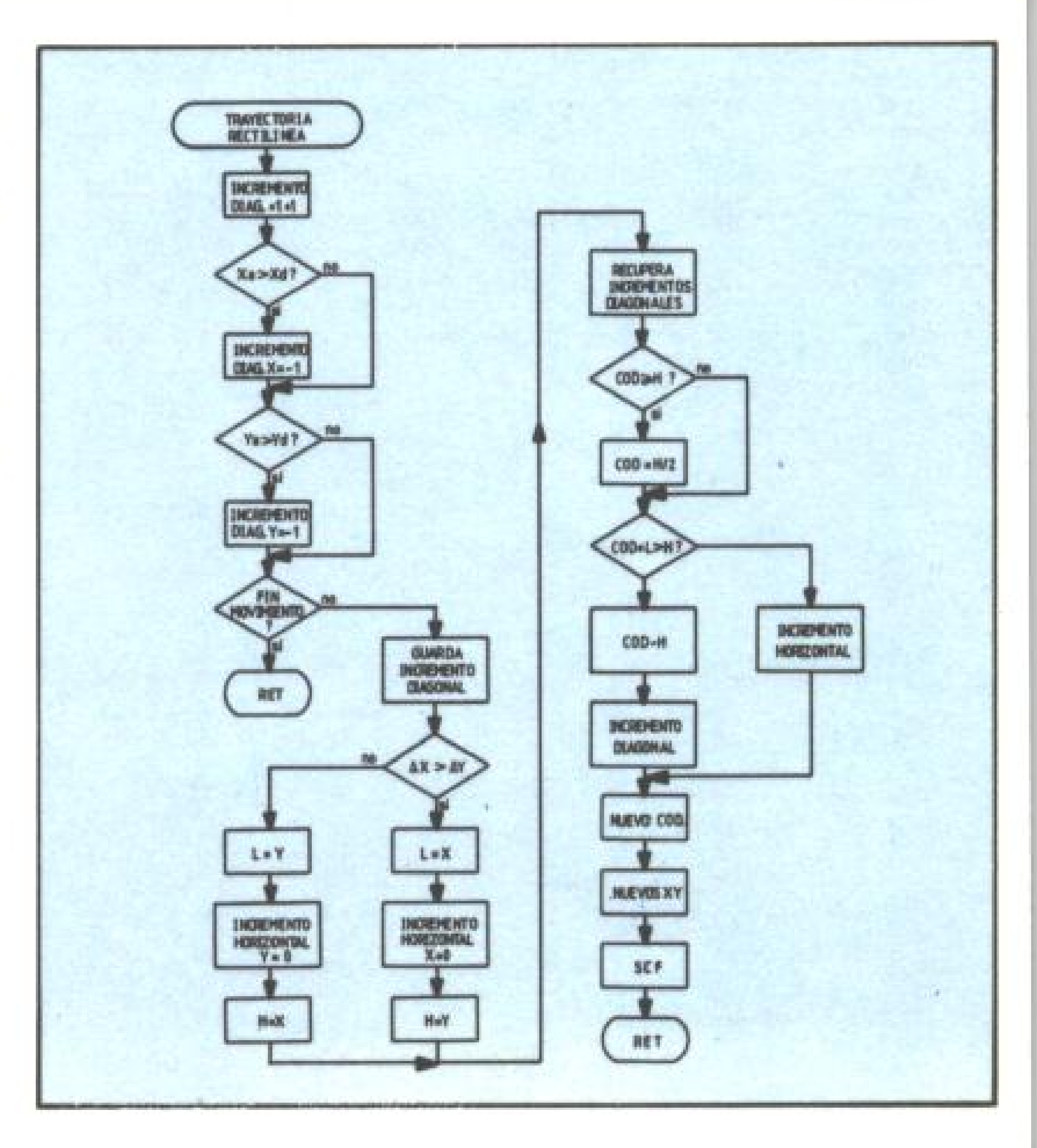
IX + 3 Coordenada X de destino.

IX + 4 Coordenada Y de destino.

A su retorno las coordenadas X e Y (+ 1 + 2) son actualizadas. Si devuelve carry es que ha habido cambios. No carry significa que el móvil llegó a su destino.

● En BASIC se conoce la llegada a destino porque la función USR devuelve 0.

IX obtiene el valor 60090 en la rutina pero puede variarse POKEando en las direcciones 60002 y 60003.



```
10 ;   **  TRAYECTORIA RECTILINEA  **
20 ;
30         ORG     60000
40 ;
50         LD      IX,60090
60 START   LD      DE,#0101 ;Presume incs +1 +1
70         LD      A,(IX+3) ;X destino
80         SUB     (IX+1)   ;X actual
90         JR      NC,INCS1 ;Salta si Xact<Xdest
100        LD      E,#FF    ;Incremento X = -1
110        NEG     ;        ;A = ABS (Xdest-Xact)
120 INCS1  LD      C,A      ;C=Inc abs X
130        LD      A,(IX+4) ; Y destino
140        SUB     (IX+2)   ; Y actual
150        JR      NC,INCS2 ;Salta si Yact<Ydest
160        LD      D,#FF    ;Incremento Y = -1
170        NEG     ;        ;A = ABS (Ydest-Yact)
180 ;
190 INCS2  LD      B,A      ;B=Inc abs Y
200        OR      C        ;Test inc X e inc Y=0
210        RET     Z;       ;Si esta en el destino
220        PUSH    DE       ;Incrementos diagonal
230        LD      A,B      ;B=Inc abs Y
240        CP      C        ;Inc absoluto X
250        JR      NC,INCS3 ;Salta si incX<incY
260        LD      L,B      ;L=incremento de Y
270        LD      D,0      ;Inc. horizontal Y
280        JR      INCS4
290 INCS3  LD      L,C      ;L=incremento de X
300        LD      C,B      ;C=Inc Y
310        LD      E,0      ;Inc. horizontal X
320 INCS4  LD      H,C      ;H=Maximo (incx,incy)
330        POP     BC       ;Incrementos diagonal
340 ;
350        LD      A,(IX+0) ;Codigo anterior
360        CP      H        ;Incremento mayor
370        JR      C,INCS5  ;No debe superar
380        LD      A,H      ; al incremento
390        SRL     A        ;Si cod>=H, A=H/2
400 INCS5  ADD     A,L      ;Suma inc. menor
410        JR      C,DIAG   ;Es mayor que H
420        CP      H        ;Si es mayor que H
430        JR      C,HOR    ; desp. horizontal.
440 ;
450 DIAG   SUB     H        ;Resta H al codigo
460        LD      D,B      ;Pasa inc. diagonal
470        LD      E,C      ; al par DE
480 HOR    LD      (IX+0),A ;Nuevo codigo
490        LD      A,E      ;Incremento de X
500        ADD     A,(IX+1) ;Lo suma a X actual
510        LD      (IX+1),A ;Siguiente X
520        LD      A,D      ;Incremento de Y
530        ADD     A,(IX+2) ;Lo suma a Y actual
540        LD      (IX+2),A ;Siguiente Y
550        SCF     ;        ;No estaba en
560        RET     ;        ; el destino.
```

```
10 DATA "DD 21 BA EA 11 01 01 DD",914
20 DATA "7E 03 DD 96 01 30 04 1E",583
30 DATA "FF ED 44 4F DD 7E 04 DD",1211
40 DATA "96 02 30 04 16 FF ED 44",786
50 DATA "47 B1 C8 D5 78 B9 30 05",1019
60 DATA "68 16 00 18 04 69 48 1E",361
70 DATA "00 61 C1 DD 7E 00 BC 38",881
80 DATA "03 7C CB 3F 85 38 03 BC",773
90 DATA "38 03 94 50 59 DD 77 00",716
100 DATA "7B DD 86 01 DD 77 01 7A",942
110 DATA "DD 86 02 DD 77 02 37 C9",955
120 DATA "                       ",0
```

Música I

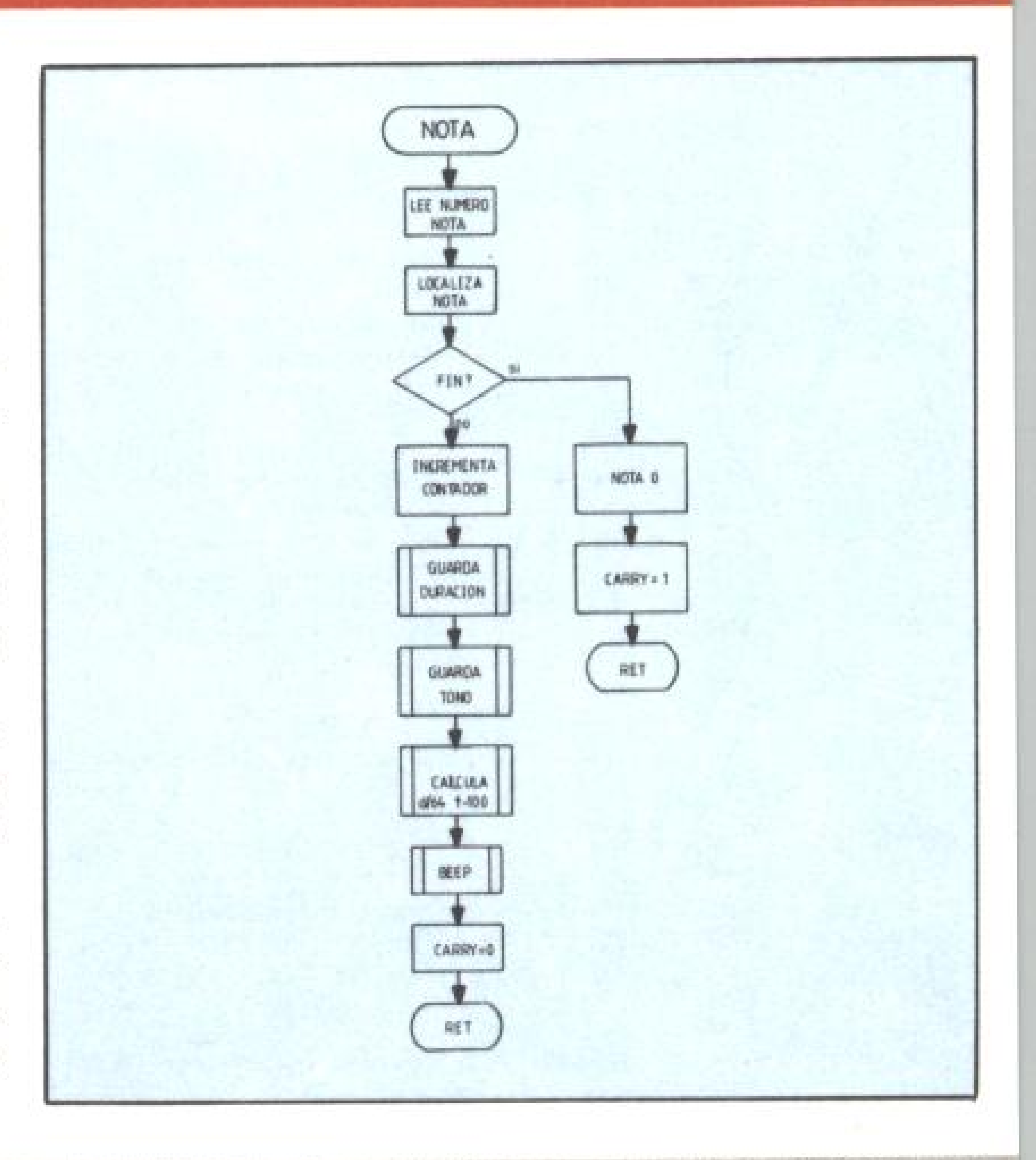
Dos fichas comprenden las rutinas de música que ofrecemos.

El comando **BEEP** necesita dos valores para su funcionamiento. Estos pueden ser fraccionarios e incluso negativos por lo que los datos de una sola nota ocupan entre 15 y 20 bytes si están en BASIC y al menos 10 mientras los almacenemos en formato de coma flotante.

El sistema que proponemos es multiplicar la duración por 64 y sumar al tono 100. De esta forma con sólo dos bytes podremos almacenar cualquier nota de la redonda a la semifusa y en 10 octavas.

El listado BASIC que acompañamos se encarga de crear este formato que se compone de una cabecera de 2 bytes, un cuerpo de 2 bytes por nota y un byte marca de final (255).

La rutina en código máquina (USR 60000) ejecuta una nota incrementando el puntero o poniéndolo a 0 si detecta la señal de fin de melodía. Esta rutina necesita para su funcionamiento las que aparecerán en la ficha (MUSICA II).



```
 10 ;  **   MUSICA    - I -    **
 20 ;
 30         ORG     60000
 40 ;
 50 ;   *  TOCA UNA NOTA   *
 60 ;
 70 NOTA    LD      HL,17      ;Direccion musica.
 80         PUSH    HL
 90         POP     DE         ;La copia en DE.
100         LD      C,(HL)     ;Lee numero de nota*2.
110         INC     HL
120         LD      B,(HL)
130         INC     HL
140         ADD     HL,BC      ;Localiza la nota.
150         LD      A,(HL)     ;Lee primer dato.
160         EX      DE,HL      ;HL =direcc. partitura.
170         CP      #FF        ;Si el dato no es FF
180         JR      NZ,CONT    ; toca la nota.
190         XOR     A          ;Si es FF nota 0
200         LD      (HL),A
210         INC     HL
220         LD      (HL),A
230         SCF     ;          ;Senal fin partitura.
240         RET
250 ;
260 CONT    INC     BC         ;Siguiente nota.
270         INC     BC
280         LD      (HL),C     ;Carga direccion
290         INC     HL
300         LD      (HL),B     ; de la nota siguente.
310         EX      DE,HL
320         PUSH    HL
330         CALL    STAKA      ;Guarda duracion
340         POP     HL         ; en el stk del calc.
350         INC     HL
360         LD      A,(HL)     ;Guarda tono en el
370         CALL    STAKA      ;Stack del calculador.
380 ;
390         RST     #28        ;Calculador.
400         DEFB    EX,NUM,#40,#B0,0,64 ;Numero 64.
410         DEFB    DIV,EX     ;Duracion/64; Tono.
420         DEFB    NUM,#40,#B0,0,100 ;Numero 100.
430         DEFB    REST,END ;Resta 100 al tono.
440         CALL    BEEP       ;Toca la nota.
450         XOR     A          ;Senal no fin part.
460         RET
```

```
 10 LET dir=61000
 20 LET long=8
 30 POKE dir,0
 40 POKE dir+1,0
 50 FOR n=1 TO long
 60 READ d,t: BEEP d,t
 70 POKE dir+2*n,d*64
 80 POKE dir+2*n+1,t+100
 90 NEXT n
100 POKE dir+2*n,255
110 DATA 1,0,1,2,.5,3,.5,2
120 DATA 1,0,1,3,-1,5,2,7
```

Esta segunda parte de rutinas de música no funciona sin la primera aparecida en la ficha anterior de esta serie. No son reubicables.

El listado de DATAs que acompaña corresponde a ambas partes conjuntamente.

Utilización

— Inicialización de una melodía:

BASIC RANDOMIZE dir : LET M = USR 60088
CM LD DE,direc CALL START

— Ejecutar una nota (la siguiente):

BASIC LET M = USR 60000
CM CALL NOTA (Carry sin fin melodía).

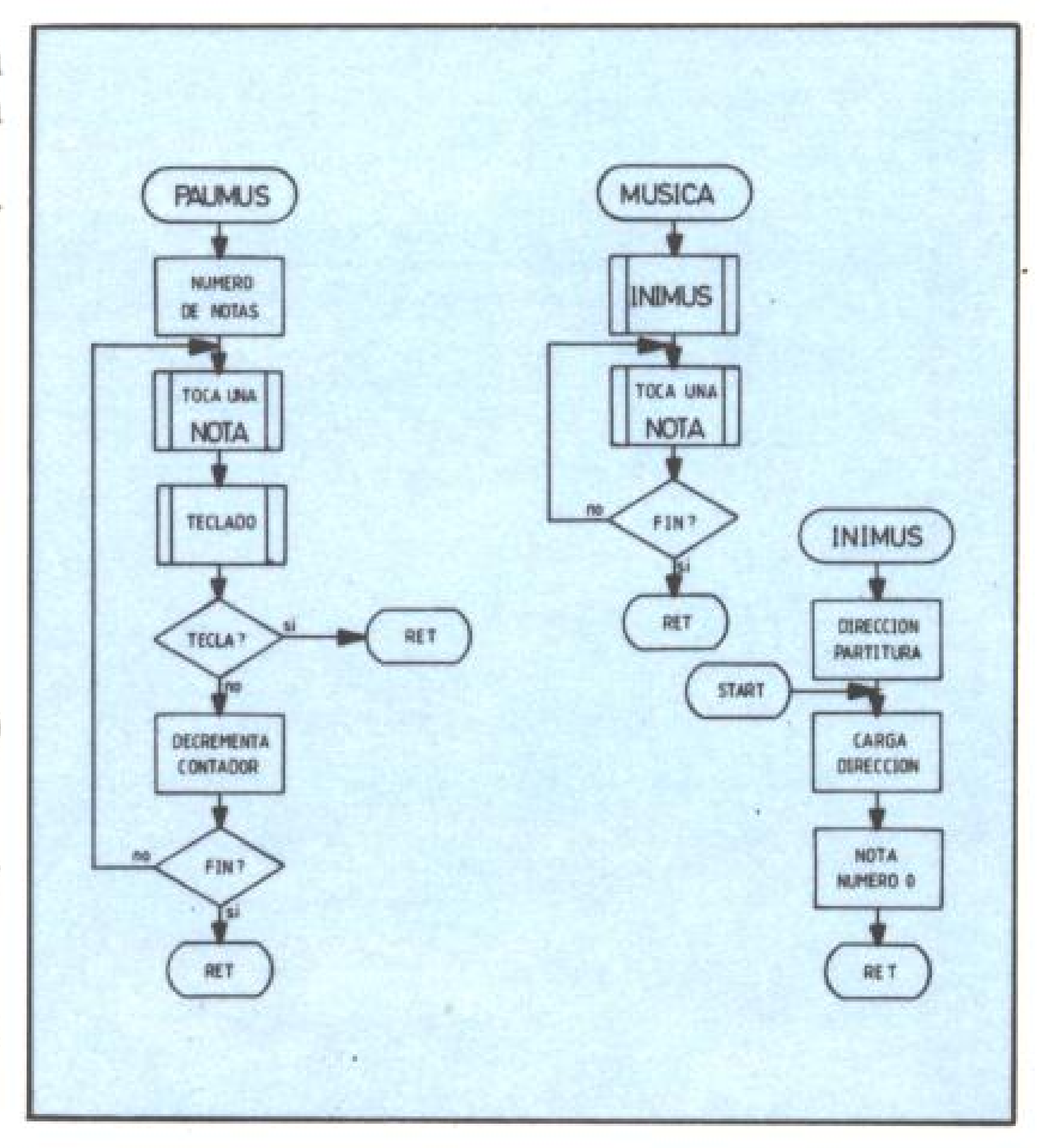
— Ejecutar una melodía:

BASIC RANDOMIZE dir : LET M = USR 60079
CM LD DE,direc CALL INIMUS
CALL BUCMUS

(Puede cambiarse 660 INIMUS por START y funcionará LD DE,dir CALL MUSICA)

— Pausa musical:

BASIC RANDOMIZE n : LET M = USR 60059
CM LD BC,notas CALL BUCPM



```
470 ; **   MUSICA   - II -   **
480 ;
490 ;  * PAUSA CON MUSICA *
500 ;
510 PAUMUS LD     BC,(SEED);Num. dado en RANDOMIZE
520 BUCPM  PUSH   BC
530        CALL   NOTA     ;Toca una nota.
540        CALL   KEYSCN   ;Consulta el teclado.
550        INC    E        ;Si E=#FF no tecla.
560        POP    BC
570        RET    NZ       ;Si se pulso tecla.
580        DEC    BC       ;Decrementa contador.
590        LD     A,B
600        OR     C
610        JR     NZ,BUCPM ;Continua si no es 0.
620        RET
630 ;
640 ;   * TOCA UNA MELODIA *.
650 ;
660 MUSICA CALL   INIMUS   ;Inicializa partitura.
670 BUCMUS CALL   NOTA     ;Toca una nota.
680        RET    C        ;Fin partitura.
690        JR     BUCMUS   ;Siguiente nota.
700 ;
710 ; * INICIALIZA UNA MELODIA *
720 ;
730 INIMUS LD     DE,(SEED);Act. por RANDOMIZE.
740 START  LD     HL,NOTA+1
750        LD     (HL),E   ;Carga la direccion.
760        INC    HL
770        LD     (HL),D   ;En NOTA+1 y +2
780        XOR    A
790        LD     (DE),A   ;Nota numero 0.
800        INC    DE
810        LD     (DE),A
820        RET
```

```
830 ;
840 ;
850 KEYSCN EQU    #28E     ;Consulta el teclado.
860 BEEP   EQU    #3F8
870 STAKA  EQU    #2D28    ;Pasa A al stack del
880 ;                     ;Calculador.
890 ;
900 SEED   EQU    23670    ;Act. por RANDOMIZE.
910 ;
920 REST   EQU    #3       ;Resta.
930 DIV    EQU    #5       ;Division.
940 NUM    EQU    #34      ;Prefijo de numero.
950 EX     EQU    #1       ;Intercambia datos.
960 END    EQU    #38      ;Fin de calculos.
```

```
 10 DATA "21 11 00 E5 D1 4E 23 46",671
 20 DATA "23 09 7E EB FE FF 20 06",952
 30 DATA "AF 77 23 77 37 C9 03 03",710
 40 DATA "71 23 70 EB E5 CD 28 2D",1014
 50 DATA "E1 23 7E CD 28 2D EF 01",916
 60 DATA "34 40 B0 00 40 05 01 34",414
 70 DATA "40 B0 00 64 03 38 CD F8",852
 80 DATA "03 AF C9 ED 4B 76 5C C5",1098
 90 DATA "CD 60 EA CD 8E 02 1C C1",1105
100 DATA "C0 0B 78 B1 20 F1 C9 CD",1179
110 DATA "B8 EA CD 60 EA D8 18 FA",1443
120 DATA "ED 5B 76 5C 21 61 EA 73",1017
130 DATA "23 72 AF 12 13 12 C9   ",580
```

Función gráfica I

Esta rutina nos permite dibujar la gráfica de una función con la ampliación o reducción que se desee. Es reubicable.

La función gráfica se define:

DEF FN G(F$,X,L,Y,M) = USR 60000

En ella F$ representa a F(x)

X y L: límites mínimo y máximo de X.

Y y M: límites mínimo y máximo de y.

La función gráfica dibujará la función matemática y nos devolverá el punto que corresponde al eje de la Y, (X = 0).

Ejemplo:

PLOT FN G("0",—10,10, —2,2),0: DRAW 0,175: RANDOMIZE FN G ("SIN X",—10,10, —2,2) nos dibujará los ejes de coordenadas y la función seno entre los límites —10 > = x = < 10 , —2 > = y = < 2.

Nota: Debido a su longitud esta rutina continúa en la ficha siguiente.

```
;        ***  FUNCION GRAFICA ***
;
        ORG     60000
;
START   RES     0,(IY+2) ;Parte sup. pantalla.
        CALL    TEMPS    ;Asigna atributos
        LD      HL,(CHADD)
        PUSH    HL       ;Guarda CHADD
        LD      HL,(DEFADD);Direccion de DEF FN
        LD      B,5      ;   Las 5 variables
BUCSTK  PUSH    BC
        INC     HL       ;     saltando nombre
        INC     HL       ;     y CHR$ 14
        INC     HL       ;     son pasadas
        CALL    STKNUM   ;     al stack
        POP     BC       ;     del calculador.
        DJNZ    BUCSTK
;
        LD      HL,MEMORY;     Memoria auxiliar
        LD      (MEM),HL
        RST     #28      ;Calculador;F$,X,L,Y,M
        DEFB    EX       ;F$,X,L,M,Y
        DEFB    #C0      ;F$,X,L,M,Y ; MEM0=Y
        DEFB    SUB      ;F$,X,L,M-Y
        DEFB    NUM,#40,#B0,0,175; Guarda 175
        DEFB    EX       ;F$,X,L,175,M-Y
        DEFB    DIV      ;175/Y-M = Inc Y
        DEFB    #C1      ;F$,X,L,IncY;MEM1=IncY
        DEFB    #E0      ;F$,X,L,IncY,Y
        DEFB    MUL      ;F$,X,L,IncY*Y=BaseY
        DEFB    #C0      ;MEM0 = Base Y
        DEFB    DEL      ;F$,X,L
        DEFB    EX       ;F$,L,X
        DEFB    #C3      ;F$,L,X ; MEM3=X
        DEFB    SUB      ;F$,L-X
        DEFB    NUM,#40,#B0,0,255;Guarda 255
        DEFB    DIV      ;F$,L-X/255=IncX
        DEFB    EX       ;IncX,F$
        DEFB    END      ;Fin de los calculos.
```

```
400 ;
410         LD      HL,MEMBOT; Mem. ordinaria
420         LD      (MEM),HL
430         LD      B,VAL     ;Funcion VAL
440         RST     #28       ;Calculador.
450         DEFB    VAL       ;IncX,VAL F$=F(x)
460         DEFB    END       ;Fin de los calculos.
470 ;
480         LD      BC,0      ;256 puntos X
490 BUCLE   PUSH    BC        ;Guarda contador
500         CALL    BREAK     ;Test de BREAK
510         JP      NC,ERRORL;Si BREAK error L
520 ;
530         LD      HL,MEMORY;    Memoria auxiliar.
540         LD      (MEM),HL
550         RST     #28       ;    Calculador.
560         DEFB    #E1       ;IncX,F(x),IncY
570         DEFB    MUL       ;IncX,F(x)*IncY
580         DEFB    #E0       ;IncX,F(x)*IncY,BaseY
590         DEFB    SUB       ;IncX,F(X)*IncY-BaseY
600 ;                       ; = PlotY
610         DEFB    #C2       ;MEM2 = PlotY
620         DEFB    NEG?      ;IncX,(1/0)
630         DEFB    SRV,NOPLOT-$ ;  Si< 0 no pinta.
640         DEFB    #E2       ;IncX,PlotY
650         DEFB    NUM,#40,#B0,0,175;Guarda 175
660         DEFB    SUB       ;IncX,PlotY-175
670         DEFB    POS?      ;IncX,(1,0)
680         DEFB    SRV,NOPLOT-$;Si>175 tampoco pinta.
690         DEFB    #E2,END   ;IncX,PlotY.
700 ;
710         CALL    FPTOA     ;A=Alto de la pila.
720         LD      B,A       ;B=Coord Y
730         POP     AF        ;A=Contador
740         PUSH    AF        ;Lo repone.
750         NEG     ;         ;256-Contador =
760         LD      C,A       ; coord X
770         CALL    PLOTSB    ;Dibuja punto
780         RST     #28       ;Calculador.
```

```
 10 DATA "FD CB 02 86 CD 4D 0D 2A",929
 20 DATA "5D 5C E5 2A 0B 5C 06 05",570
 30 DATA "C5 23 23 23 CD B4 33 C1",931
 40 DATA "10 F6 21 1A EB 22 68 5C",786
 50 DATA "EF 01 C0 03 34 40 B0 00",727
 60 DATA "AF 01 05 C1 E0 04 C0 02",796
 70 DATA "01 C3 03 34 40 B0 00 FF",746
 80 DATA "05 01 38 21 92 5C 22 68",471
 90 DATA "5C 06 1D EF 1D 38 01 00",452
100 DATA "00 C5 CD 54 1F D2 7B 1B",877
110 DATA "21 1A EB 22 68 5C EF E1",988
120 DATA "04 E0 03 C2 36 00 1A E2",731
130 DATA "34 40 B0 00 AF 03 37 00",525
140 DATA "10 E2 38 CD D5 2D 47 F1",1073
150 DATA "F5 ED 44 4F CD E5 22 EF",1336
160 DATA "38 ED 5B 0B 5C 21 0B 00",531
170 DATA "19 22 68 5C EF 31 E0 0F",782
180 DATA "C0 02 38 21 92 5C 22 68",659
190 DATA "5C 2A 61 5C 22 5D 5C CD",747
200 DATA "FB 24 C1 10 AC E1 22 5D",1020
210 DATA "5C 21 1A EB 22 68 5C EF",855
220 DATA "02 E3 A0 01 03 01 05 38",455
230 DATA "CD A2 2D 38 01 C8 01 00",670
240 DATA "00 C9 00 00 00 00 00 00",201
250 DATA "00 00 00 00 00 00 00 00",0
260 DATA "00 00 00 00 00 00      ",0
```

Función gráfica II

En esta ficha se encuentra la segunda parte y última de la rutina Función gráfica.

Funcionamiento

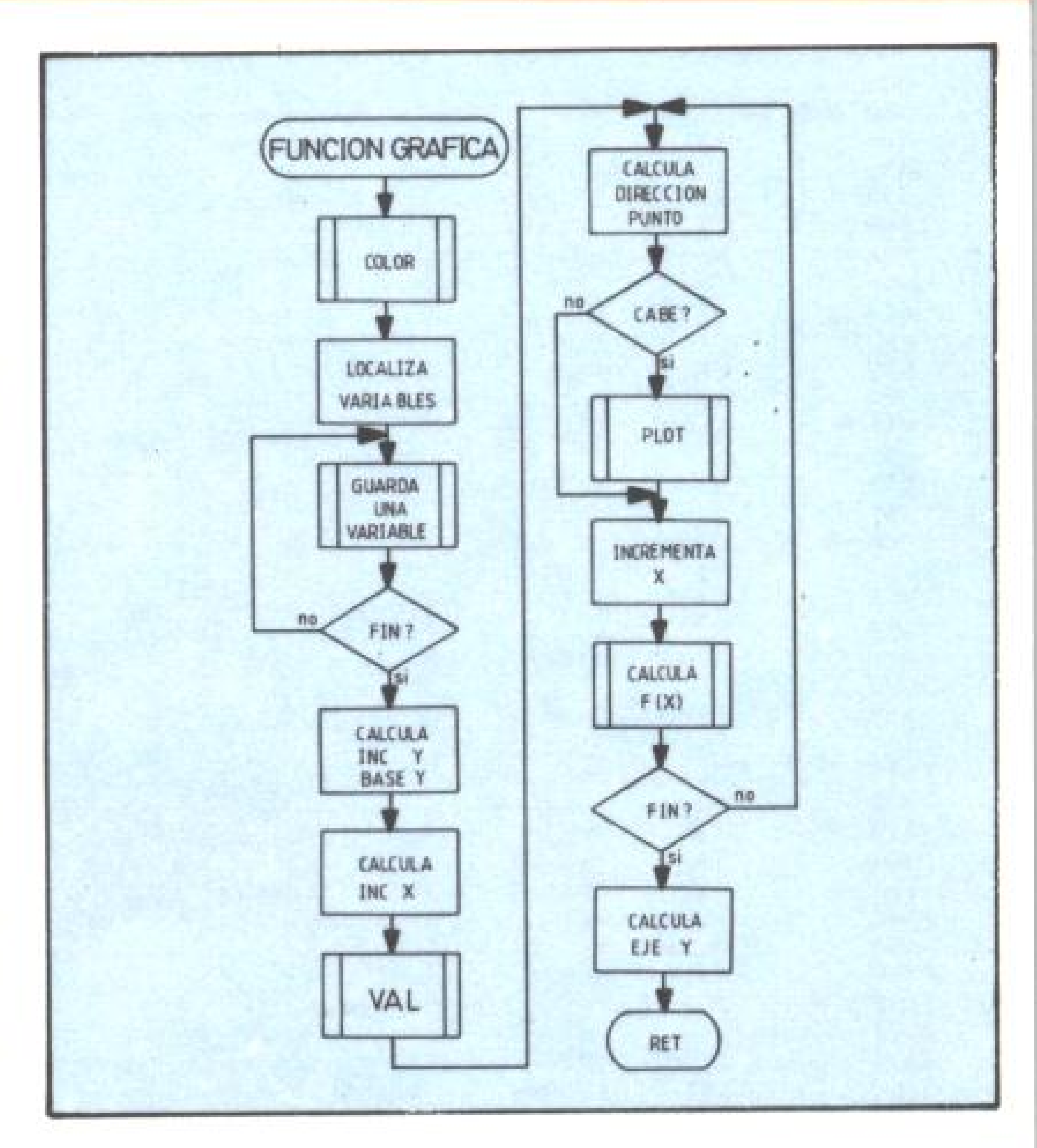
Al principio pone la bandera de utilización de la parte superior de la pantalla y llama a TEMPS para asignar el color.

El bucle BUCSTK guarda una a una las 5 variables de que consta la función.

Se efectúa la función VAL para pasar la función al espacio de trabajo y hallar el primer valor de F(x).

El BUCLE principal comprueba si se ha pulsado BREAK, calcula las coordenadas del punto, lo dibuja si se encuentra dentro de los límites y averigua de nuevo el valor de la función para el punto siguiente. Para esto último se usa SCANNING en lugar de VAL pues es mucho más rápida.

Por último calcula la dirección en pantalla del eje de las Y, el correspondiente a X = 0.

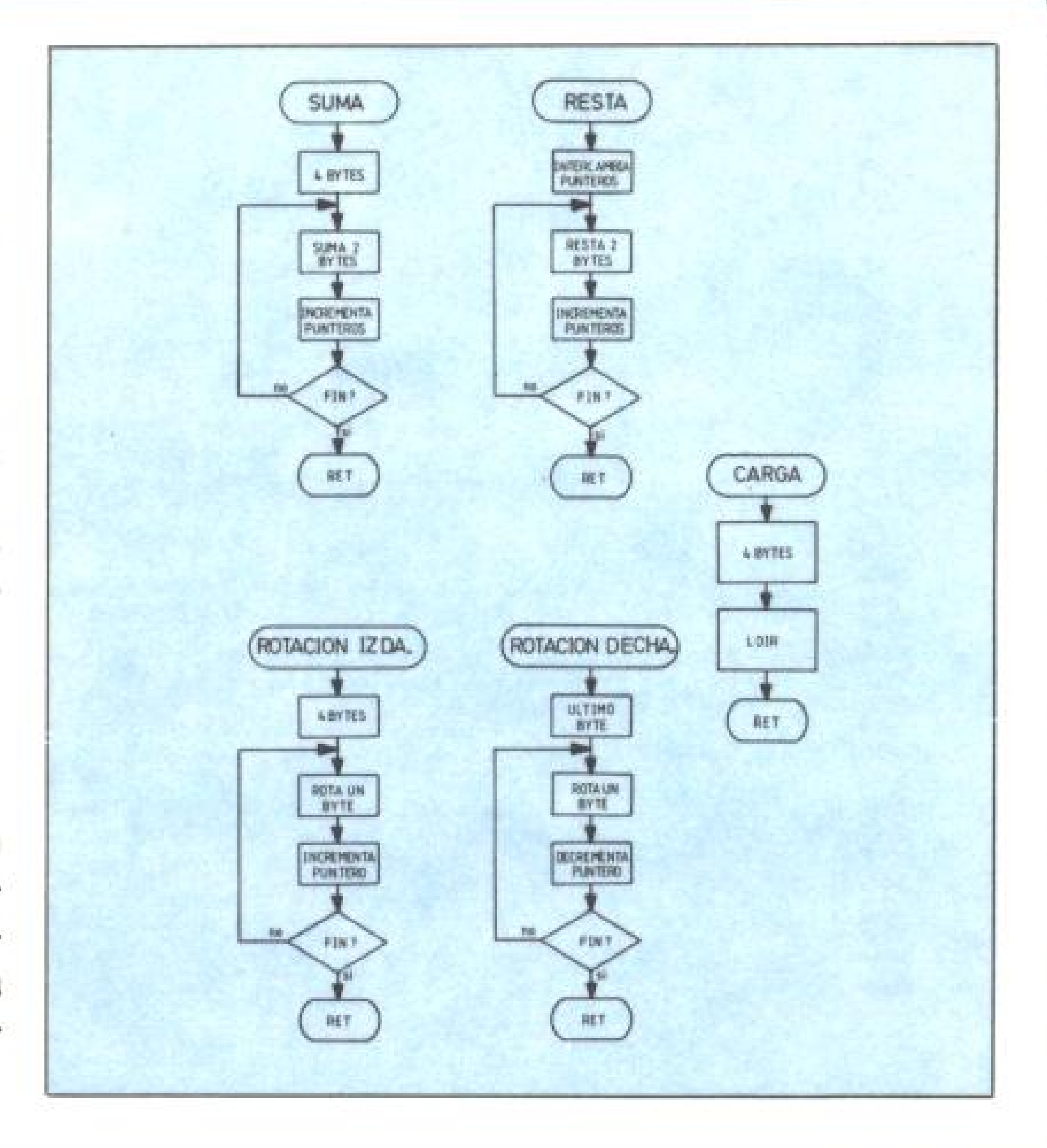


```
 790 ;
 800 NOPLOT DEFB   END        ;Salida si no pinta.
 810        LD     DE,(DEFADD);Direcc. funcion.
 820        LD     HL,11      ; Variable X
 830        ADD    HL,DE      ; como MEM provisional
 840        LD     (MEM),HL
 850        RST    #28        ;Calculador.
 860        DEFB   DUP        ;IncX,IncX
 870        DEFB   #E0        ;IncX,IncX,X
 880        DEFB   SUM        ;IncX,IncX+X=Nueva X
 890        DEFB   #C0        ;IncX,X ; X=Nueva X
 900        DEFB   DEL        ;IncX
 910        DEFB   END        ;Fin de los calculos.
 920 ;
 930        LD     HL,MEMBOT; Mem. ordinaria
 940        LD     (MEM),HL
 950        LD     HL,(WORKSP); VAL F$
 960        LD     (CHADD),HL
 970        CALL   SCAN       ;Nuevo F(X)
 980        POP    BC         ;Recupera contador.
 990        DJNZ   BUCLE      ;Nuevo punto
1000 ;
1010        POP    HL         ;Recupera CHADD
1020        LD     (CHADD),HL
1030        LD     HL,MEMORY; Memoria auxiliar
1040        LD     (MEM),HL
1050        RST    #28        ;Calculador;IncX,F(x)
1060        DEFB   DEL        ;IncX
1070        DEFB   #E3        ;IncX,X inicial.
1080        DEFB   #A0        ;IncX,X,0
1090        DEFB   EX         ;IncX,0,X
1100        DEFB   SUB        ;IncX,0-X
1110        DEFB   EX         ;0-X,IncX
1120        DEFB   DIV        ;(0-X)/IncX=Coor.Y
1130        DEFB   END        ;Fin de los calculos.
1140        CALL   FPTOBC     ; A = BC = Eje Y
1150        JR     C,FUERA    ;Es mayor de 255
1160        RET    Z          ;Numero positivo.
1170 ;
1180 FUERA  LD     BC,0       ;Si fuera hacerlo 0.
1190        RET               ; Final de la rutina.
1200 ;
1210 MEMORY DEFS   20         ;Memoria auxiliar.
1220 ;
1230 CHADD  EQU    23645      ;Puntero interprete.
1240 DEFADD EQU    23563      ;Direccion DEF FN
1250 MEM    EQU    23656      ;Puntero memoria.
1260 MEMBOT EQU    23698      ;Memoria ordinaria.
1270 WORKSP EQU    23649      ;Espacio de trabajo.
1280 ;
1290 TEMPS  EQU    #0D4D      ;Asigna color
1300 STKNUM EQU    #33B4      ;Pasa num. al STK
1310 BREAK  EQU    #1F54      ;Test de BREAK
1320 ERRORL EQU    #1B7B      ;Mensaje error L.
1330 FPTOA  EQU    #2DD5      ;Alto del STK a A.
1340 FPTOBC EQU    #2DA2      ;Alto del STK a BC.
1350 PLOTSB EQU    #22E5      ;Dibuja un punto.
1360 SCAN   EQU    #24FB      ;Evalua expresion.
1370 ;
1380 SRV    EQU    0          ;Salto rel. si verdad.
1390 NEG?   EQU    #36        ;Es menor que 0?
1400 POS?   EQU    #37        ;Es mayor que 0?
1410 SUM    EQU    #0F        ;Suma
1420 SUB    EQU    #03        ;Resta
1430 MUL    EQU    #04        ;Multiplica
1440 DIV    EQU    #05        ;Divide
1450 DUP    EQU    #31        ;Repite el dato
1460 EX     EQU    #01        ;Cambia 2 datos
1470 DEL    EQU    #02        ;Elimina dato
1480 VAL    EQU    #1D        ;Funcion VAL
1490 NUM    EQU    #34        ;Prefijo numero
1500 END    EQU    #38        ;Fin calculador.
```

32 bits I: Suma, Resta, Rotación, Carga

Esta es la primera ficha del grupo que tratará de aritmética de 32 bits. Estas rutinas ofrecen la posibilidad de operar con números muy grandes, siendo mucho más rápidas que las de coma flotante que usa el calculador de la ROM del Spectrum.

En esta ficha ofrecemos, además de las de suma y resta, dos rutinas de rotación a derecha e izquierda con carry, ampliaciones de RR y RL, que serán útiles para multiplicar, dividir y otras operaciones más complejas. Por último la rutina de carga, que implementa la instrucción "LD" para 32 bits.



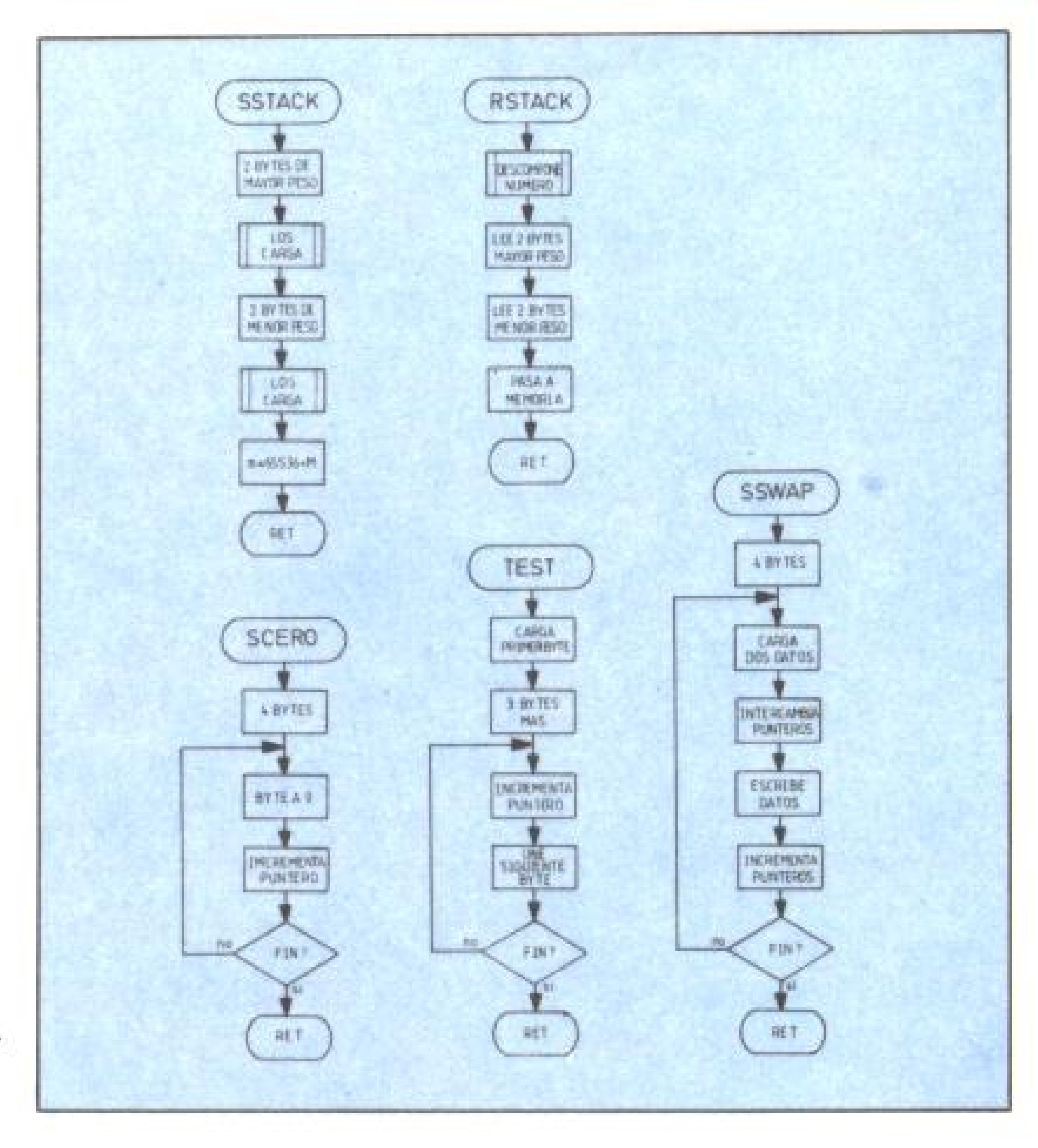
Utilización:

Los datos que utilizan estas rutinas deberán situarse en una zona especial de variables para 32 bits. Estas pueden ser fácilmente localizables si las usamos numeradas, pues basta multiplicar su número por 4 para conocer su lugar.

```
10 ;   ** CALCULO 32 BITS  - 1 - **
20 ;
30 ;
40 ;SUMA (HL)=(HL)+(DE)
50 ;
60 SSUMA  LD     B,4       ;Opera con 4 bytes.
70        OR     A         ;Carry a 0.
80 XSM    LD     A,(DE)
90        ADC    A,(HL)    ;Suma a (HL) el (DE)
100       LD     (HL),A    ; y guarda la suma
110 ;                      en en el segundo.
120       INC    DE        ;Punt. primer sumando.
130       INC    HL        ;Puntero del segundo
140 ;                      sumando y resultado.
150       DJNZ   XSM       ;Siguiente byte.
160       RET
170 ;
180 ;
190 ;RESTA (HL)=(HL)-(DE)
200 ;
210 SREST LD     B,4       ;Opera con 4 bytes.
220       EX     DE,HL     ;Intercamb. registros
230 XRS   LD     A,(DE)
240       SBC    A,(HL)    ;Resta (DE) a (HL)
250       LD     (DE),A    ;  y guarda el resul.
260       INC    DE        ;Punt. del minuendo
270 ;                      y resultado.
280       INC    HL        ;Punt. del sutraendo.
290       DJNZ   XRS       ;Siguiente byte.
300       RET
310 ;
320 ;
330 ;ROTACION A LA IZQUIERDA  CON CARRY DE (HL)
340 ;
350 SRIZQ LD     B,4       ;Opera con 4 bytes.
360 XRIZQ RL     (HL)      ;Rota un byte.
370       INC    HL        ;Incrementa puntero.
380       DJNZ   XRIZQ     ;Siguiente byte.
390       RET
400 ;
410 ;
420 ;ROTACION A LA DERECHA CON CARRY DE (HL)
430 ;
440 SRDCH LD     B,4       ;Opera con 4 bytes.
450       INC    HL
460       INC    HL        ;Puntero en el
470       INC    HL        ;  ultimo byte.
480 XRDCH RR     (HL)      ;Rota un byte.
490       DEC    HL        ;Decrementa puntero.
500       DJNZ   XRDCH     ;Byte anterior.
510       RET
520 ;
530 ;
540 ;CARGA (DE) CON (HL)
550 ;
560 ;NO AFECTA AL CARRY
570 ;
580 SMOVE LD     BC,4      ;4 bytes por copiar.
590       LDIR   ;         ;Los copia.
600       RET
```

32 bits II: Stk, 0, Test, Swap

En ciertos momentos puede ser necesario el intercambio de datos entre el stack del calculador y las variables de 32 bits. Las dos primeras rutinas ofrecen esa posibilidad.



Funcionamiento:

Para guardar un número en el stack del calculador pasa primero la parte de menos peso y luego la más significativa, después con la rutina del calculador se multiplica la de mayor peso por 65536 y se suma a la de menor peso.

Para el proceso inverso se usa la rutina 32H del calculador (N mod M) que descompone un número en dos partes.

Otras tres rutinas completan la ficha:

Una pone a 0 los cuatro bytes de una variable.

La siguiente comprueba si una variable es 0, devolviendo el resultado en el flag Z.

Y la última sirve para intercambiar los valores de dos variables.

```
610 ;   **  CALCULO 32 BITS  - II -   **
620 ;
630 ;CARGA (HL) EN EL STACK DEL CALCULADOR
640 ;
650 SSTACK LD      C,(HL)    ;Carga los dos
660        INC     HL        ; bytes menos
670        LD      B,(HL)    ; significativos
680        INC     HL        ; en el stack
690        PUSH    HL
700        CALL    STKBC     ; del calclador.
710        POP     HL        ;Recupera puntero.
720        LD      C,(HL)    ;Ahora guarda los
730        INC     HL        ; dos bytes mas
740        LD      B,(HL)    ; significativos
750        CALL    STKBC     ; en el stack del
760        RST     #28       ; calculador.
770        DEFB    #34,0,#41,0 ;Guarda 65536
780        DEFB    4         ;N(menos sig.)*65536
790        DEFB    #F        ;N(m.s.)*65536+N(M.s.)
800        DEFB    #38       ;Fin de los calculos.
810        RET
820 ;
830 ;PASA  A   (HL) EL NUMERO DE LO ALTO
840 ;DEL STACK DEL CALCULADOR
850 ;
860 RSTACK PUSH    HL        ;Guarda puntero.
870        RST     #28       ;Calculador.
880        DEFB    #34,0,#41,0;Guarda 65536
890        DEFB    #32,#38   ;Lo descompone.
900        CALL    FPTOBC    ;Parte mas signif.
910        PUSH    BC        ;La guarda.
920        CALL    FPTOBC    ;Parte menos sig.
930        POP     DE        ;Parte mas sig.
940        POP     HL        ;Recupera puntero.
950        LD      (HL),C
960        INC     HL
970        LD      (HL),B    ;Carga los
980        INC     HL
990        LD      (HL),E    ; cuatro bytes.
```

```
1000         INC     HL
1010         LD      (HL),D
1020         RET
1030 ;
1040 ; HACE (HL)=0
1050 ;
1060 SCERO   LD      B,4        ;Numero de bytes.
1070 BUC0    LD      (HL),0     ;Pone a 0 un byte.
1080         INC     HL         ;Incrementa contador.
1090         DJNZ    BUC0       ;Siguiente byte.
1100         RET
1110 ;
1120 ;TEST (HL)=0
1130 ;
1140 SEQ0    LD      A,(HL)     ;Primer byte.
1150         LD      B,3        ;Tres bytes mas.
1160 XEQ0    INC     HL         ;Incrementa puntero.
1170         OR      (HL)       ;Une el sig. byte.
1180         DJNZ    XEQ0       ;Siguiente byte.
1190         RET     ;          ;Z si todos son 0.
1200 ;                          ;NZ si alguno no es 0.
1210 ;
1220 ;INTERCAMBIO ENTRE (HL) Y (DE)
1230 ;
1240 ;NO AFECTA AL CARRY
1250 ;
1260 SSWAP   LD      B,4        ;Son 4 bytes
1270 SSWAB   LD      A,(DE)     ;Carga los datos
1280         LD      C,(HL)     ; en A y C
1290         EX      DE,HL      ;Cambia punteros.
1300         LD      (DE),A     ;Carga los datos
1310         LD      (HL),C     ; intercambiados.
1320         INC     HL         ;Incrementa
1330         INC     DE         ; los punteros.
1340         DJNZ    SSWAB      ;Siguienta byte.
1350         RET
1360 ;
1370 STKBC   EQU     #2D2B      ;Pasa BC al stack.
1380 FPTOBC  EQU     #2DA2      ;Lee num. del stack.
```

32 bits III: Multiplicación, División

Para poder utilizar estas rutinas se necesitan las que aparecen en las dos fichas anteriores pues son utilizadas por éstas.

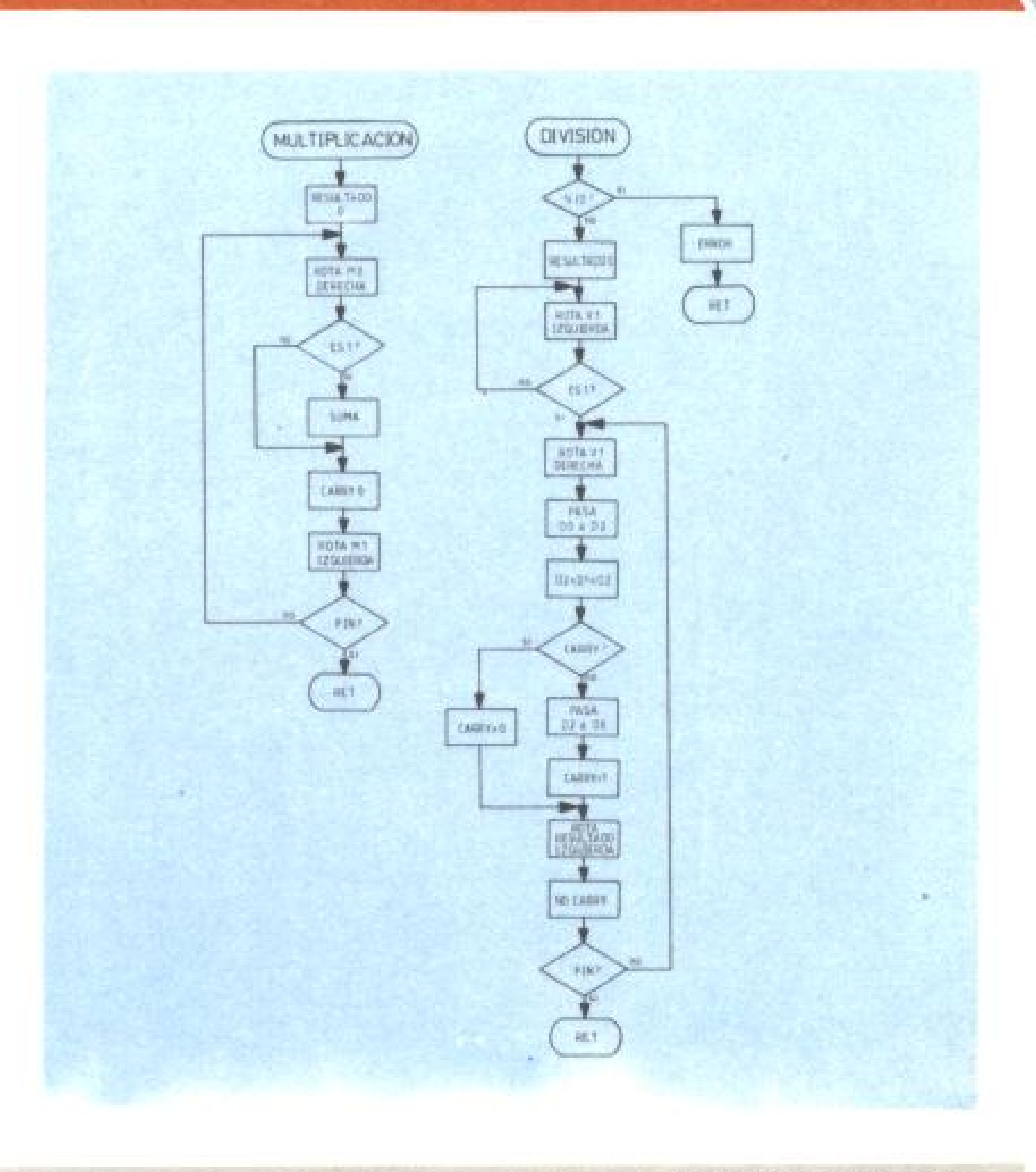
Multiplicación:

Los bits que componen el multiplicador son extraídos por la derecha. Si el bit encontrado es 1 se suma el multiplicador al resultado parcial y si es 0 no.

Cada vez el multiplicador es duplicado (rotado a la izquierda) para, de esta forma, ser sumado al nuevo resultado parcial.

División:

En primer lugar se localiza la primera cifra significativa por la izquierda, su posición determinará el número de cifras del resultado. Estas cifras van entrando por la izquierda siendo 0 ó 1 según el resultado de la resta del dividendo y el divisor desplazado (un bit cada ciclo).



```
1390 ;**   CALCULO 32 BITS  - III -   **
1400 ;
1410 ;
1420 ;
1430 VMR     DEFS   4          ;Producto.
1440 VM0     DEFS   4          ;Multiplicando.
1450 VM1     DEFS   4          ;Multiplicador.
1460 ;
1470 ;
1480 VDR     DEFS   4          ;Cociente.
1490 VD0     DEFS   4          ;Dividendo.
1500 VD1     DEFS   4          ;Divisor.
1510 VD2     DEFS   4          ;Auxiliar division.
1520 ;
1530 ;
1540 ;
1550 ;MULTIPLICACION VMR=VM0*VM1    *
1560 ;
1570 ; CARRY DESCONOCIDO
1580 ;
1590 SMULT  LD     HL,VMR     ;Inicializa con 0
1600        CALL   SCERO      ; el resultado.
1610        LD     B,32       ;Hay 32 bits.
1620 BUCBIT PUSH   BC         ;Guarda contador.
1630        OR     A          ;Carry = 0.
1640        LD     HL,VM0     ;Multiplicando.
1650        CALL   SRDCH      ;Obtiene un bit.
1660        JR     NC,CONTM   ;Si es 0 no suma.
1670        LD     HL,VMR     ;Si es 1 suma
1680        LD     DE,VM1     ;  el multiplicador
1690        CALL   SSUMA      ;  al resultado.
1700 CONTM  OR     A          ;Carry = 0.
1710        LD     HL,VM1     ;El multiplicador una
1720        CALL   SRIZQ      ; cifra a la izquierda.
1730        POP    BC         ;Contador de bits.
1740        DJNZ   BUCBIT     ;Siguiente bit.
1750        RET
1760 ;
1770 ;
```

```
1780 ;DIVISON VDR=VD0/VD1
1790 ;CARRY A 1 SI SE DIVIDE    ENTRE 0
1800 ;
1810 SDIV   LD     HL,VD1     ;Si el divisor
1820        CALL   SEQ0       ; es igual a 0
1830        JP     Z,ERROR    ; no se puede dividir.
1840        LD     HL,VDR     ;Se inicializa el
1850        CALL   SRES       ; resultado con 0.
1860        LD     B,0        ;Contador de bits.
1870 XD1    INC    B          ;Incrementa contador.
1880        LD     HL,VD1     ;Rota el divisor
1890        PUSH   BC
1900        CALL   SRIZQ      ; a la izquierda
1910        POP    BC         ; hasta la primera
1920        JR     NC,XD1     ; cifra significativa.
1930 XD5    PUSH   BC         ;Guarda contador.
1940        LD     HL,VD1     ;El divisor se
1950        CALL   SRDCH      ; rota a la derecha.
1960        LD     HL,VD0     ;Copia la var. 0
1970        LD     DE,VD2     ; en la var. 2
1980        CALL   SMOVE
1990        LD     HL,VD2
2000        LD     DE,VD1
2010        CALL   SREST      ;Resta Var 2 - Var 1.
2020        JR     C,XD0
2030        LD     HL,VD2
2040        LD     DE,VD0     ;Copia Var 2 en Var 0.
2050        CALL   SMOVE
2060        SCF    ;           Carry = 1.
2070        JR     XD4
2080 XD0    OR     A          ;Carry = 0.
2090 XD4    LD     HL,VDR     ;Rota el resultado
2100        CALL   SRIZQ      ; a la izquierda
2110        POP    BC         ; anadiendo bit.
2120        XOR    A          ;Carry = 0.
2130        DJNZ   XD5        ;Continua el bucle.
2140        RET
2150 ERROR  SCF    ;           Carry = 1.
2160        RET
```

Interrupciones - Doble Borde

Las rutinas de esta ficha permiten la ejecución de cualquier rutina durante las interrupciones enmascarables:

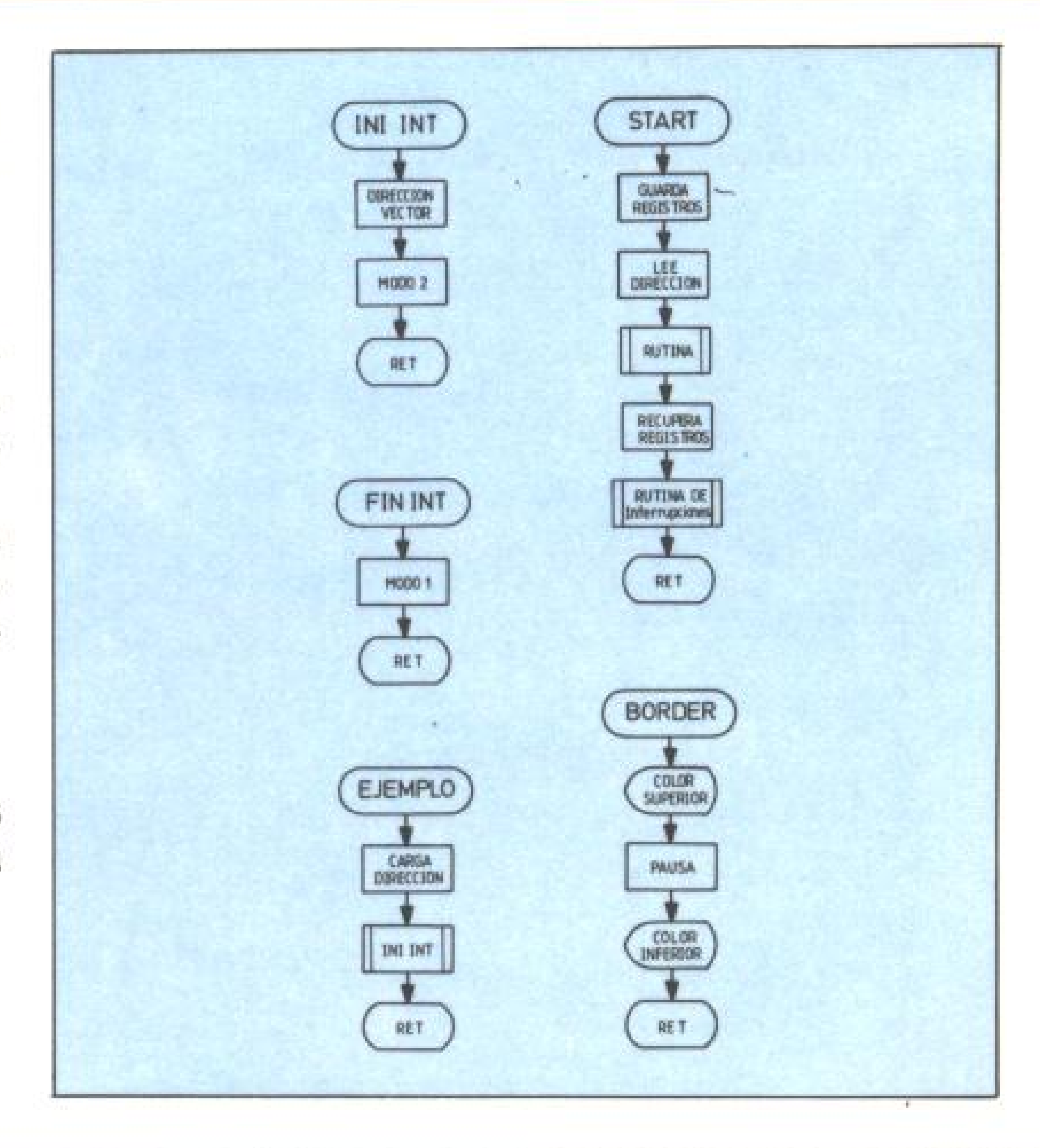
- INIINT, (65230) activa el mecanismo.
- FININT, (65237) lo desactiva.
- (START) guarda todos los registros, ejecuta la subrutina deseada, recupera los registros y finaliza saltando a la rutina ordinaria de interrupciones.

Para iniciar el funcionamiento de la rutina que deseemos se debe cargar en los bytes 65277-65278 (DIRINT) la dirección de ésta y, posteriormente, llamar a INIINT (65230).

Doble Borde:

Como ejemplo de utilización de estas rutinas ofrecemos una rutina que muestra en pantalla un Borde de 2 colores.

Inicialización	:	65281
Ajuste de altura	:	65298,65299
Color superior	:	65291
Color inferior	:	65302



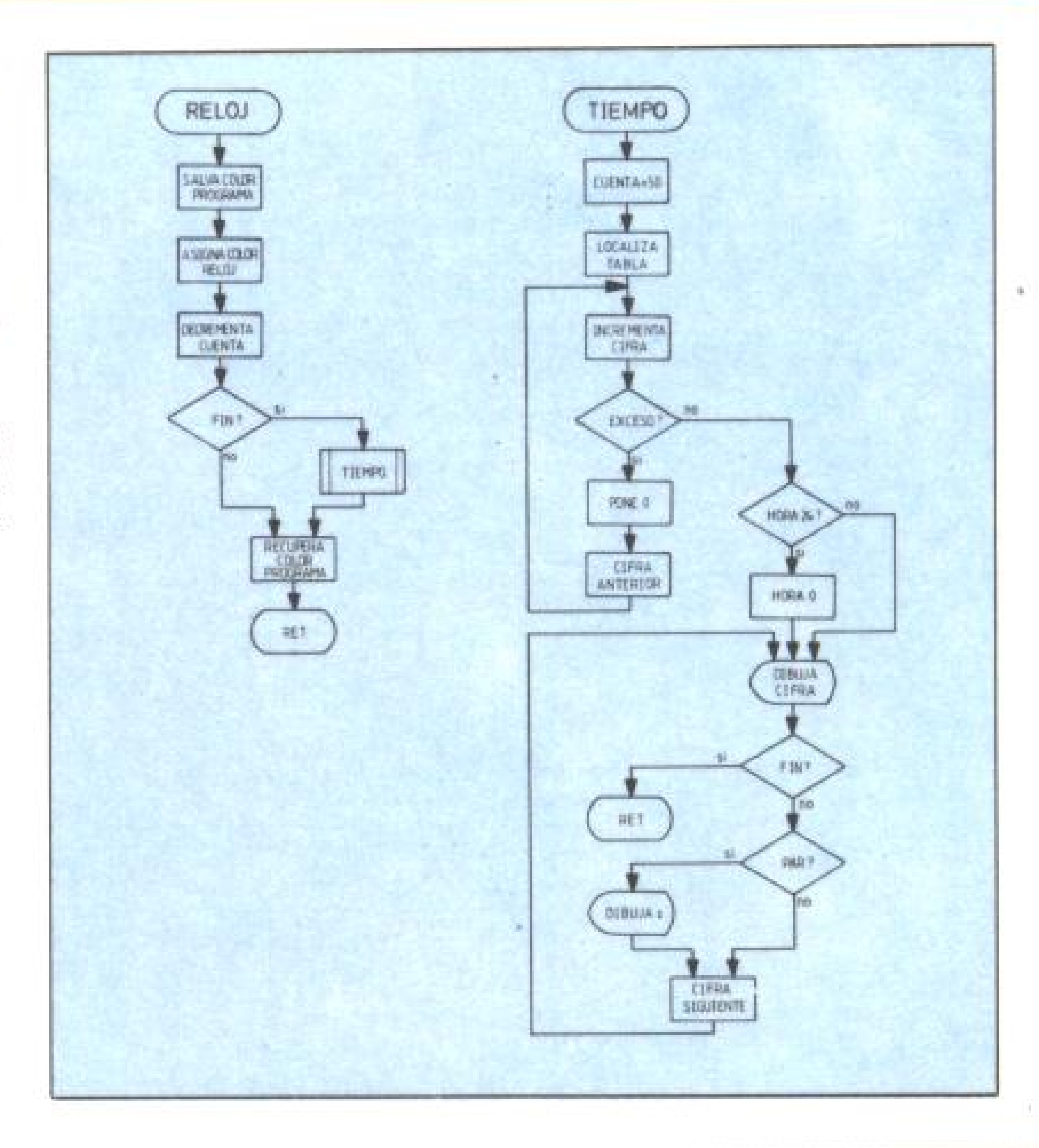
```
 10 *****   I N T E R R U P C I O N E S  ******
 20 ;
 30         ORG     65230
 40 ;
 50 INIINT  LD      A,#FE      ;Parte alta de la
 60         LD      I,A        ; direccion de "INTER"
 70         IM      2          ; (la baja es FFH).
 80         RET
 90 ;
100 FININT  IM      1          ;RUTINA DE DESACTIVACION
110         RET
120 ;
130 START   PUSH    AF
140         PUSH    BC         ;Guarda los registros
150         PUSH    DE         ; ordinarios.
160         PUSH    HL
170         PUSH    IX
180         PUSH    IY
190 ;
200         EXX     ;          ;Intercambia los
210         EX      AF,AF'     ; registros alternativos.
220 ;
230         PUSH    AF
240         PUSH    BC         ;Guarda los registros
250         PUSH    DE         ; alternativos.
260         PUSH    HL
270 ;
280         LD      HL,(DIRINT);Carga dir. rutina.
290         CALL    #162C      ;La ejecuta:"JP (HL)".
300 ;
310         POP     HL
320         POP     DE         ;Recupera registros
330         POP     BC         ; alternativos.
340         POP     AF
350 ;
360         EX      AF,AF'     ;Intercambia registros
370         EXX     ;           ordinarios.
380 ;
390         POP     IY
400         POP     IX
410         POP     HL         ;Recupera registros
420         POP     DE         ; ordinarios.
430         POP     BC
440         POP     AF
450 ;
460         JP      #38        ;Interrupcion ordinaria.
470 ;
480 DIRINT  DEFS    2          ;Direccion rutina.
490 INTER   DEFW    START      ;Direccion del vector
500 ;                          ; de interrupciones.
510 ;
520 ; ****  E J E M P L O  ****
530 ;
540 EJEMP   LD      HL,BORDER;Direccion rutina.
550         LD      (DIRINT),HL
560         CALL    INIINT     ;Activa el sistema.
570         RET
580 ;
590 BORDER  LD      A,5        ;Color superior.
600         OUT     (#FE),A    ;Lo pinta.
610         LD      D,H        ;DE=HL para no modificar
620         LD      E,L        ; la memoria con LDIR.
630         LD      BC,1523    ;Altura del color.
640         LDIR    ;          ;Pausa.
650         LD      A,4        ;Color inferior.
660         OUT     (#FE),A    ;Lo pinta.
670         RET
```

```
 10 DATA "3E FE ED 47 ED 5E C9 ED",1393
 20 DATA "56 C9 F5 C5 D5 E5 DD E5",1621
 30 DATA "FD E5 D9 08 F5 C5 D5 E5",1591
 40 DATA "2A FD FE CD 2C 16 E1 D1",1254
 50 DATA "C1 F1 08 D9 FD E1 DD E1",1583
 60 DATA "E1 D1 C1 F1 C3 38 00 64",1219
 70 DATA "FE D8 FE 21 0B FF 22 FD",1310
 80 DATA "FE CD CE FE C9 3E 05 D3",1398
 90 DATA "FE 54 5D 01 F3 05 ED B0",1093
100 DATA "3E 04 D3 FE C9          ",732
```

Reloj

Se podrá visualizar un reloj en la pantalla al mismo tiempo que se ejecuta otro programa, salvo en el caso de que éste deshabilite las interrupciones. Por este motivo el reloj se parará durante la ejecución del comando BEEP.

Esta rutina debe estar acompañada de la que aparece en la ficha «INTERRUPCIONES» (M-35). Puede hacerse el volcado de DATAS bajo esta última en la dirección 65114 (no es reubicable) y salvarlas conjuntamente mediante SAVE «nombre» CODE 65114,167.



Utilización

Poner en marcha: Randomize USR 65114
Parar : Randomize USR 65237
Cambiar color : POKE 65129,8*papel + tinta.
Poner en hora : INPUT «HHMMSS»; t$: FOR n = 1TO 6: POKE 65224 + n,CODE t$(n)
12 horas:POKE 65170,49:POKE 65176,61: POKE 65180,49
24 horas:POKE 65170,50:POKE 65176,52: POKE 65180,48

```
10 ;*** RELOJ ***
20 ;
30         ORG    65114
40 ;
50         LD     HL,RELOJ ;Direccion rutina.
60         LD     (DIRINT),HL
70         CALL   INIINT   ;Activa el sistema.
80         RET
90 ;
100 RELOJ  LD     HL,(#5C8F);Salva ATTRT y
110        PUSH   HL        ;           MASK-T
120        LD     HL,#000F ;Papel 1, tinta 7
130        LD     (#5C8F),HL;Lo carga en ATTRT.
140        LD     HL,CUENTA
150        DEC    (HL)       ;1 segundo son
160        CALL   Z,TIEMPO
170        POP    HL         ;Recupera ATTRT y
180        LD     (#5C8F),HL;          MASK-T
190        RET
200 ;
210 TIEMPO LD     (HL),50
220        LD     DE,TMAX+5;Final tabla maximos
230        LD     HL,HMS+5 ;Final tabla tiempo
240 INCRE  LD     A,(DE)   ;Maximo
250        INC    (HL)      ;Incrementa dato
260        SUB    (HL)      ;Si no es mayor que
270        JR     NC,FIN    ; el maximo termina.
280        LD     (HL),"0" ;Lo pone a 0 e
290        DEC    HL        ; inc.el siguiente.
300        DEC    DE        ;Maximo siguiente
310        JR     INCRE     ;Proxima cifra.
320 FIN    LD     HL,HMS    ;Hora
330        LD     A,(HL)    ;Si la cifra alta
340        CP     "2"       ; es un 2
350        JR     NZ,PRINT ;continua
360        INC    HL        ; Si es un 2 pero
370        LD     A,(HL)    ; la cifra baja
380        CP     "4"       ; no es un 4
390        JR     NZ,PRINT ; tambien continua
400        LD     (HL),"0" ; La hora 24
410        DEC    HL
420        LD     (HL),"0" ; es la hora 0
430 PRINT  LD     BC,#1809 ;Linea 0 col. 24
440        LD     HL,16384+24;Direc. pantalla
450        LD     DE,HMS    ;Puntero caracteres
460 BUC    PUSH   DE        ;Lo guarda
470        LD     A,(DE)    ;Codigo de la cifra
480        CALL   #0B65     ;POCHAR;Imp.caracter.
490        POP    DE        ;Recupera puntero
500        LD     A,L       ;Columna
510        CP     32        ;Si es la ultima
520        RET    NC        ;Fin escritura
530        BIT    0,E       ;Si es cifra par
540        JR     Z,CONT    ; continua
550        LD     A,":"     ; separador
560        PUSH   DE        ;Puntero a la cifra
570        CALL   #0B65     ;POCHAR;Imp.separador.
580        POP    DE        ;Recupera puntero
590 CONT   INC    DE        ;Siguiente cifra
600        JR     BUC
610 CUENTA DEFB   1         ;Contador interrup.
620 TMAX   DEFM   "295959"  ;Tabla de maximos
630 HMS    DEFM   "000000"  ;Cuadro del reloj
640 INIINT EQU    65230
650 DIRINT EQU    65277
```

```
10 DATA "21 64 FE 22 FD FE CD CE",1339
20 DATA "FE C9 2A 8F 5C E5 21 0F",1009
30 DATA "00 22 8F 5C 21 C1 FE 35",802
40 DATA "CC 7A FE E1 22 8F 5C C9",1275
50 DATA "36 32 11 C7 FE 21 CD FE",1066
60 DATA "1A 34 96 30 06 36 30 2B",427
70 DATA "1B 18 F5 21 C8 FE 7E FE",1163
80 DATA "32 20 0B 23 7E FE 34 20",592
90 DATA "05 36 30 2B 36 30 01 09",262
100 DATA "18 21 18 40 11 C8 FE D5",829
110 DATA "1A CD 65 0B D1 7D FE 20",963
120 DATA "D0 CB 43 28 07 3E 3A D5",858
130 DATA "CD 65 0B D1 13 18 E8 01",802
140 DATA "32 39 35 39 35 39 30 30",423
150 DATA "30 30 30 30             ",192
```